

activemq-cpp-3.1.0

Generated by Doxygen 1.7.1

Mon Jan 24 2011 16:15:22

Contents

1	Namespace Index	1
1.1	Namespace List	1
2	Data Structure Index	3
2.1	Class Hierarchy	3
3	Data Structure Index	21
3.1	Data Structures	21
4	File Index	51
4.1	File List	51
5	Namespace Documentation	69
5.1	activemq Namespace Reference	69
5.1.1	Detailed Description	69
5.2	activemq::cmsutil Namespace Reference	70
5.3	activemq::commands Namespace Reference	71
5.4	activemq::core Namespace Reference	72
5.5	activemq::exceptions Namespace Reference	73
5.6	activemq::io Namespace Reference	73
5.7	activemq::library Namespace Reference	73
5.8	activemq::state Namespace Reference	73
5.9	activemq::threads Namespace Reference	74
5.10	activemq::transport Namespace Reference	74
5.11	activemq::transport::correlator Namespace Reference	75
5.12	activemq::transport::failover Namespace Reference	75
5.13	activemq::transport::inactivity Namespace Reference	76
5.14	activemq::transport::logging Namespace Reference	76
5.15	activemq::transport::mock Namespace Reference	76
5.16	activemq::transport::tcp Namespace Reference	77

5.17	activemq::util Namespace Reference	77
5.18	activemq::wireformat Namespace Reference	78
5.19	activemq::wireformat::openwire Namespace Reference	78
5.20	activemq::wireformat::openwire::marshal Namespace Reference	78
5.21	activemq::wireformat::openwire::marshal::v1 Namespace Reference	79
5.22	activemq::wireformat::openwire::marshal::v2 Namespace Reference	83
5.23	activemq::wireformat::openwire::marshal::v3 Namespace Reference	87
5.24	activemq::wireformat::openwire::marshal::v4 Namespace Reference	90
5.25	activemq::wireformat::openwire::marshal::v5 Namespace Reference	94
5.26	activemq::wireformat::openwire::utils Namespace Reference	98
5.27	activemq::wireformat::stomp Namespace Reference	99
5.28	cms Namespace Reference	99
5.28.1	Detailed Description	102
5.29	decaf Namespace Reference	102
5.29.1	Detailed Description	102
5.30	decaf::internal Namespace Reference	102
5.31	decaf::internal::io Namespace Reference	103
5.32	decaf::internal::net Namespace Reference	103
5.33	decaf::internal::nio Namespace Reference	103
5.34	decaf::internal::util Namespace Reference	104
5.35	decaf::internal::util::concurrent Namespace Reference	104
5.36	decaf::io Namespace Reference	105
5.37	decaf::lang Namespace Reference	106
5.37.1	Function Documentation	107
5.37.1.1	operator!=	107
5.37.1.2	operator!=	107
5.37.1.3	operator==	107
5.37.1.4	operator==	107
5.38	decaf::lang::exceptions Namespace Reference	108
5.39	decaf::net Namespace Reference	108
5.40	decaf::nio Namespace Reference	109
5.41	decaf::security Namespace Reference	110
5.42	decaf::security::auth Namespace Reference	110
5.43	decaf::security::auth::x500 Namespace Reference	110
5.44	decaf::security::cert Namespace Reference	110
5.45	decaf::security_provider Namespace Reference	111

5.46	decaf::security_provider::unix Namespace Reference	111
5.47	decaf::security_provider::unix::openssl Namespace Reference	111
5.48	decaf::util Namespace Reference	112
5.49	decaf::util::comparators Namespace Reference	114
5.50	decaf::util::concurrent Namespace Reference	114
5.51	decaf::util::concurrent::atomic Namespace Reference	115
5.52	decaf::util::concurrent::locks Namespace Reference	116
5.53	decaf::util::logging Namespace Reference	116
5.53.1	Enumeration Type Documentation	117
5.53.1.1	Level	117
5.54	std Namespace Reference	117
6	Data Structure Documentation	119
6.1	decaf::util::AbstractCollection< E > Class Template Reference	119
6.1.1	Detailed Description	121
6.1.2	Constructor & Destructor Documentation	122
6.1.2.1	~AbstractCollection	122
6.1.3	Member Function Documentation	122
6.1.3.1	add	122
6.1.3.2	addAll	122
6.1.3.3	clear	123
6.1.3.4	contains	124
6.1.3.5	containsAll	124
6.1.3.6	copy	125
6.1.3.7	equals	125
6.1.3.8	isEmpty	125
6.1.3.9	lock	126
6.1.3.10	notify	126
6.1.3.11	notifyAll	126
6.1.3.12	operator=	127
6.1.3.13	remove	127
6.1.3.14	removeAll	128
6.1.3.15	retainAll	128
6.1.3.16	toArray	129
6.1.3.17	tryLock	129
6.1.3.18	unlock	129
6.1.3.19	wait	130

6.1.3.20	wait	130
6.1.3.21	wait	131
6.1.4	Field Documentation	131
6.1.4.1	mutex	131
6.2	decaf::util::AbstractList< E > Class Template Reference	131
6.2.1	Detailed Description	132
6.2.2	Constructor & Destructor Documentation	132
6.2.2.1	~AbstractList	132
6.3	decaf::util::AbstractMap< K, V, COMPARATOR > Class Template Reference	132
6.3.1	Detailed Description	133
6.3.2	Constructor & Destructor Documentation	133
6.3.2.1	~AbstractMap	133
6.4	decaf::util::AbstractQueue< E > Class Template Reference	133
6.4.1	Detailed Description	134
6.4.2	Constructor & Destructor Documentation	135
6.4.2.1	AbstractQueue	135
6.4.2.2	~AbstractQueue	135
6.4.3	Member Function Documentation	135
6.4.3.1	add	135
6.4.3.2	addAll	135
6.4.3.3	clear	136
6.4.3.4	element	136
6.4.3.5	remove	136
6.5	decaf::util::AbstractSequentialList< E > Class Template Reference	137
6.5.1	Detailed Description	137
6.5.2	Constructor & Destructor Documentation	138
6.5.2.1	~AbstractSequentialList	138
6.6	decaf::util::AbstractSet< E > Class Template Reference	138
6.6.1	Detailed Description	138
6.6.2	Constructor & Destructor Documentation	139
6.6.2.1	~AbstractSet	139
6.6.3	Member Function Documentation	139
6.6.3.1	removeAll	139
6.7	activemq::transport::AbstractTransportFactory Class Reference	139
6.7.1	Detailed Description	140
6.7.2	Constructor & Destructor Documentation	140

6.7.2.1	<code>~AbstractTransportFactory</code>	140
6.7.3	Member Function Documentation	140
6.7.3.1	<code>createWireFormat</code>	140
6.8	<code>activemq::core::ActiveMQAckHandler</code> Class Reference	141
6.8.1	Detailed Description	141
6.8.2	Constructor & Destructor Documentation	141
6.8.2.1	<code>~ActiveMQAckHandler</code>	141
6.8.3	Member Function Documentation	141
6.8.3.1	<code>acknowledgeMessage</code>	141
6.9	<code>activemq::commands::ActiveMQBlobMessage</code> Class Reference	142
6.9.1	Constructor & Destructor Documentation	143
6.9.1.1	<code>ActiveMQBlobMessage</code>	143
6.9.1.2	<code>~ActiveMQBlobMessage</code>	143
6.9.2	Member Function Documentation	143
6.9.2.1	<code>clone</code>	143
6.9.2.2	<code>cloneDataStructure</code>	143
6.9.2.3	<code>copyDataStructure</code>	143
6.9.2.4	<code>equals</code>	144
6.9.2.5	<code>getDataStructureType</code>	144
6.9.2.6	<code>getMimeType</code>	144
6.9.2.7	<code>getName</code>	144
6.9.2.8	<code>getRemoteBlobUrl</code>	145
6.9.2.9	<code>isDeletedByBroker</code>	145
6.9.2.10	<code>setDeletedByBroker</code>	145
6.9.2.11	<code>setMimeType</code>	145
6.9.2.12	<code>setName</code>	145
6.9.2.13	<code>setRemoteBlobUrl</code>	146
6.9.2.14	<code>toString</code>	146
6.9.3	Field Documentation	146
6.9.3.1	<code>BINARY_MIME_TYPE</code>	146
6.9.3.2	<code>ID_ACTIVEMQBLOBMESSAGE</code>	146
6.10	<code>activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller</code> Class Reference	146
6.10.1	Detailed Description	147
6.10.2	Constructor & Destructor Documentation	148
6.10.2.1	<code>ActiveMQBlobMessageMarshaller</code>	148
6.10.2.2	<code>~ActiveMQBlobMessageMarshaller</code>	148

6.10.3	Member Function Documentation	148
6.10.3.1	createObject	148
6.10.3.2	getDataStructureType	148
6.10.3.3	looseMarshal	148
6.10.3.4	looseUnmarshal	149
6.10.3.5	tightMarshal1	149
6.10.3.6	tightMarshal2	149
6.10.3.7	tightUnmarshal	150
6.11	activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller	
	Class Reference	150
6.11.1	Detailed Description	151
6.11.2	Constructor & Destructor Documentation	152
6.11.2.1	ActiveMQBlobMessageMarshaller	152
6.11.2.2	~ActiveMQBlobMessageMarshaller	152
6.11.3	Member Function Documentation	152
6.11.3.1	createObject	152
6.11.3.2	getDataStructureType	152
6.11.3.3	looseMarshal	152
6.11.3.4	looseUnmarshal	153
6.11.3.5	tightMarshal1	153
6.11.3.6	tightMarshal2	153
6.11.3.7	tightUnmarshal	154
6.12	activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller	
	Class Reference	154
6.12.1	Detailed Description	155
6.12.2	Constructor & Destructor Documentation	156
6.12.2.1	ActiveMQBlobMessageMarshaller	156
6.12.2.2	~ActiveMQBlobMessageMarshaller	156
6.12.3	Member Function Documentation	156
6.12.3.1	createObject	156
6.12.3.2	getDataStructureType	156
6.12.3.3	looseMarshal	156
6.12.3.4	looseUnmarshal	157
6.12.3.5	tightMarshal1	157
6.12.3.6	tightMarshal2	157
6.12.3.7	tightUnmarshal	158

6.13	activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller	
	Class Reference	158
6.13.1	Detailed Description	159
6.13.2	Constructor & Destructor Documentation	160
	6.13.2.1 ActiveMQBlobMessageMarshaller	160
	6.13.2.2 ~ActiveMQBlobMessageMarshaller	160
6.13.3	Member Function Documentation	160
	6.13.3.1 createObject	160
	6.13.3.2 getDataStructureType	160
	6.13.3.3 looseMarshal	160
	6.13.3.4 looseUnmarshal	161
	6.13.3.5 tightMarshal1	161
	6.13.3.6 tightMarshal2	161
	6.13.3.7 tightUnmarshal	162
6.14	activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller	
	Class Reference	162
6.14.1	Detailed Description	163
6.14.2	Constructor & Destructor Documentation	164
	6.14.2.1 ActiveMQBlobMessageMarshaller	164
	6.14.2.2 ~ActiveMQBlobMessageMarshaller	164
6.14.3	Member Function Documentation	164
	6.14.3.1 createObject	164
	6.14.3.2 getDataStructureType	164
	6.14.3.3 looseMarshal	164
	6.14.3.4 looseUnmarshal	165
	6.14.3.5 tightMarshal1	165
	6.14.3.6 tightMarshal2	165
	6.14.3.7 tightUnmarshal	166
6.15	activemq::commands::ActiveMQBytesMessage Class Reference	166
6.15.1	Constructor & Destructor Documentation	170
	6.15.1.1 ActiveMQBytesMessage	170
	6.15.1.2 ~ActiveMQBytesMessage	170
6.15.2	Member Function Documentation	170
	6.15.2.1 clearBody	170
	6.15.2.2 clone	170
	6.15.2.3 cloneDataStructure	170
	6.15.2.4 copyDataStructure	170

6.15.2.5	<code>equals</code>	171
6.15.2.6	<code>getBodyBytes</code>	171
6.15.2.7	<code>getBodyLength</code>	171
6.15.2.8	<code>getDataStructureType</code>	172
6.15.2.9	<code>onSend</code>	172
6.15.2.10	<code>readBoolean</code>	172
6.15.2.11	<code>readByte</code>	172
6.15.2.12	<code>readBytes</code>	173
6.15.2.13	<code>readBytes</code>	173
6.15.2.14	<code>readChar</code>	174
6.15.2.15	<code>readDouble</code>	174
6.15.2.16	<code>readFloat</code>	175
6.15.2.17	<code>readInt</code>	175
6.15.2.18	<code>readLong</code>	175
6.15.2.19	<code>readShort</code>	176
6.15.2.20	<code>readString</code>	176
6.15.2.21	<code>readUnsignedShort</code>	176
6.15.2.22	<code>readUTF</code>	177
6.15.2.23	<code>reset</code>	177
6.15.2.24	<code>setBodyBytes</code>	177
6.15.2.25	<code>toString</code>	177
6.15.2.26	<code>writeBoolean</code>	178
6.15.2.27	<code>writeByte</code>	178
6.15.2.28	<code>writeBytes</code>	178
6.15.2.29	<code>writeBytes</code>	179
6.15.2.30	<code>writeChar</code>	179
6.15.2.31	<code>writeDouble</code>	179
6.15.2.32	<code>writeFloat</code>	180
6.15.2.33	<code>writeInt</code>	180
6.15.2.34	<code>writeLong</code>	180
6.15.2.35	<code>writeShort</code>	180
6.15.2.36	<code>writeString</code>	181
6.15.2.37	<code>writeUnsignedShort</code>	181
6.15.2.38	<code>writeUTF</code>	181
6.15.3	Field Documentation	182
6.15.3.1	<code>ID_ACTIVEMQBYTESMESSAGE</code>	182

6.16	activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller	
	Class Reference	182
6.16.1	Detailed Description	183
6.16.2	Constructor & Destructor Documentation	183
	6.16.2.1 ActiveMQBytesMessageMarshaller	183
	6.16.2.2 ~ActiveMQBytesMessageMarshaller	183
6.16.3	Member Function Documentation	183
	6.16.3.1 createObject	183
	6.16.3.2 getDataStructureType	183
	6.16.3.3 looseMarshal	184
	6.16.3.4 looseUnmarshal	184
	6.16.3.5 tightMarshal1	184
	6.16.3.6 tightMarshal2	185
	6.16.3.7 tightUnmarshal	185
6.17	activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller	
	Class Reference	186
6.17.1	Detailed Description	187
6.17.2	Constructor & Destructor Documentation	187
	6.17.2.1 ActiveMQBytesMessageMarshaller	187
	6.17.2.2 ~ActiveMQBytesMessageMarshaller	187
6.17.3	Member Function Documentation	187
	6.17.3.1 createObject	187
	6.17.3.2 getDataStructureType	187
	6.17.3.3 looseMarshal	188
	6.17.3.4 looseUnmarshal	188
	6.17.3.5 tightMarshal1	188
	6.17.3.6 tightMarshal2	189
	6.17.3.7 tightUnmarshal	189
6.18	activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller	
	Class Reference	190
6.18.1	Detailed Description	191
6.18.2	Constructor & Destructor Documentation	191
	6.18.2.1 ActiveMQBytesMessageMarshaller	191
	6.18.2.2 ~ActiveMQBytesMessageMarshaller	191
6.18.3	Member Function Documentation	191
	6.18.3.1 createObject	191
	6.18.3.2 getDataStructureType	191

6.18.3.3	looseMarshal	192
6.18.3.4	looseUnmarshal	192
6.18.3.5	tightMarshal1	192
6.18.3.6	tightMarshal2	193
6.18.3.7	tightUnmarshal	193
6.19	activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller	
	Class Reference	194
6.19.1	Detailed Description	195
6.19.2	Constructor & Destructor Documentation	195
	6.19.2.1 ActiveMQBytesMessageMarshaller	195
	6.19.2.2 ~ActiveMQBytesMessageMarshaller	195
6.19.3	Member Function Documentation	195
	6.19.3.1 createObject	195
	6.19.3.2 getDataStructureType	195
	6.19.3.3 looseMarshal	196
	6.19.3.4 looseUnmarshal	196
	6.19.3.5 tightMarshal1	196
	6.19.3.6 tightMarshal2	197
	6.19.3.7 tightUnmarshal	197
6.20	activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller	
	Class Reference	198
6.20.1	Detailed Description	199
6.20.2	Constructor & Destructor Documentation	199
	6.20.2.1 ActiveMQBytesMessageMarshaller	199
	6.20.2.2 ~ActiveMQBytesMessageMarshaller	199
6.20.3	Member Function Documentation	199
	6.20.3.1 createObject	199
	6.20.3.2 getDataStructureType	199
	6.20.3.3 looseMarshal	200
	6.20.3.4 looseUnmarshal	200
	6.20.3.5 tightMarshal1	200
	6.20.3.6 tightMarshal2	201
	6.20.3.7 tightUnmarshal	201
6.21	activemq::core::ActiveMQConnection Class Reference	202
	6.21.1 Detailed Description	204
	6.21.2 Constructor & Destructor Documentation	204
	6.21.2.1 ActiveMQConnection	204

6.21.2.2	~ActiveMQConnection	205
6.21.3	Member Function Documentation	205
6.21.3.1	addDispatcher	205
6.21.3.2	addProducer	205
6.21.3.3	addTransportListener	205
6.21.3.4	close	205
6.21.3.5	createSession	206
6.21.3.6	createSession	206
6.21.3.7	destroyDestination	206
6.21.3.8	destroyDestination	207
6.21.3.9	fire	207
6.21.3.10	getClientID	207
6.21.3.11	getConnectionId	207
6.21.3.12	getConnectionInfo	208
6.21.3.13	getExceptionListener	208
6.21.3.14	getMetaData	208
6.21.3.15	isClosed	208
6.21.3.16	isStarted	208
6.21.3.17	onCommand	209
6.21.3.18	oneway	209
6.21.3.19	onException	209
6.21.3.20	removeDispatcher	209
6.21.3.21	removeProducer	209
6.21.3.22	removeSession	210
6.21.3.23	removeTransportListener	210
6.21.3.24	sendPullRequest	210
6.21.3.25	setExceptionListener	210
6.21.3.26	start	210
6.21.3.27	stop	211
6.21.3.28	syncRequest	211
6.21.3.29	transportInterrupted	211
6.21.3.30	transportResumed	211
6.22	activemq::core::ActiveMQConnectionFactory Class Reference	211
6.22.1	Constructor & Destructor Documentation	213
6.22.1.1	ActiveMQConnectionFactory	213
6.22.1.2	ActiveMQConnectionFactory	213

6.22.1.3	<code>~ActiveMQConnectionFactory</code>	213
6.22.2	Member Function Documentation	213
6.22.2.1	<code>createConnection</code>	213
6.22.2.2	<code>createConnection</code>	213
6.22.2.3	<code>createConnection</code>	214
6.22.2.4	<code>createConnection</code>	214
6.22.2.5	<code>getBrokerURL</code>	215
6.22.2.6	<code>getPassword</code>	215
6.22.2.7	<code>getUsername</code>	215
6.22.2.8	<code>setBrokerURL</code>	215
6.22.2.9	<code>setPassword</code>	216
6.22.2.10	<code>setUsername</code>	216
6.23	<code>activemq::core::ActiveMQConnectionMetaData</code> Class Reference	216
6.23.1	Detailed Description	217
6.23.2	Constructor & Destructor Documentation	217
6.23.2.1	<code>ActiveMQConnectionMetaData</code>	217
6.23.2.2	<code>~ActiveMQConnectionMetaData</code>	217
6.23.3	Member Function Documentation	217
6.23.3.1	<code>getCMSMajorVersion</code>	217
6.23.3.2	<code>getCMSMinorVersion</code>	218
6.23.3.3	<code>getCMSProviderName</code>	218
6.23.3.4	<code>getCMSVersion</code>	218
6.23.3.5	<code>getCMSXPropertyNames</code>	219
6.23.3.6	<code>getProviderMajorVersion</code>	219
6.23.3.7	<code>getProviderMinorVersion</code>	219
6.23.3.8	<code>getProviderVersion</code>	220
6.24	<code>activemq::core::ActiveMQConnectionSupport</code> Class Reference	220
6.24.1	Constructor & Destructor Documentation	222
6.24.1.1	<code>ActiveMQConnectionSupport</code>	222
6.24.1.2	<code>~ActiveMQConnectionSupport</code>	222
6.24.2	Member Function Documentation	222
6.24.2.1	<code>getClientId</code>	222
6.24.2.2	<code>getCloseTimeout</code>	222
6.24.2.3	<code>getNextLocalTransactionId</code>	223
6.24.2.4	<code>getNextSessionId</code>	223
6.24.2.5	<code>getNextTempDestinationId</code>	223

6.24.2.6	getPassword	223
6.24.2.7	getProducerWindowSize	223
6.24.2.8	getProperties	224
6.24.2.9	getSendTimeout	224
6.24.2.10	getTransport	224
6.24.2.11	getUsername	224
6.24.2.12	isAlwaysSyncSend	224
6.24.2.13	isUseAsyncSend	225
6.24.2.14	setAlwaysSyncSend	225
6.24.2.15	setClientId	225
6.24.2.16	setCloseTimeout	225
6.24.2.17	setPassword	225
6.24.2.18	setProducerWindowSize	226
6.24.2.19	setSendTimeout	226
6.24.2.20	setUseAsyncSend	226
6.24.2.21	setUsername	226
6.24.2.22	shutdownTransport	226
6.24.2.23	startupTransport	227
6.25	activemq::core::ActiveMQConstants Class Reference	227
6.25.1	Detailed Description	228
6.25.2	Member Enumeration Documentation	228
6.25.2.1	AckType	228
6.25.2.2	DestinationActions	228
6.25.2.3	DestinationOption	229
6.25.2.4	TransactionState	229
6.25.2.5	URIParam	229
6.25.3	Member Function Documentation	230
6.25.3.1	toDestinationOption	230
6.25.3.2	toString	230
6.25.3.3	toString	230
6.25.3.4	toURIOption	230
6.26	activemq::core::ActiveMQConsumer Class Reference	230
6.26.1	Constructor & Destructor Documentation	232
6.26.1.1	ActiveMQConsumer	232
6.26.1.2	~ActiveMQConsumer	232
6.26.2	Member Function Documentation	232

6.26.2.1	acknowledge	232
6.26.2.2	acknowledge	233
6.26.2.3	afterMessageIsConsumed	233
6.26.2.4	beforeMessageIsConsumed	233
6.26.2.5	clearMessagesInProgress	233
6.26.2.6	close	233
6.26.2.7	commit	234
6.26.2.8	deliverAcks	234
6.26.2.9	dequeue	234
6.26.2.10	dispatch	234
6.26.2.11	doClose	234
6.26.2.12	getConsumerId	235
6.26.2.13	getConsumerInfo	235
6.26.2.14	getLastDeliveredSequenceId	235
6.26.2.15	getMessageListener	235
6.26.2.16	getMessageSelector	235
6.26.2.17	isClosed	236
6.26.2.18	isSynchronizationRegistered	236
6.26.2.19	iterate	236
6.26.2.20	receive	236
6.26.2.21	receive	236
6.26.2.22	receiveNoWait	237
6.26.2.23	rollback	237
6.26.2.24	setLastDeliveredSequenceId	237
6.26.2.25	setMessageListener	237
6.26.2.26	setSynchronizationRegistered	237
6.26.2.27	start	238
6.26.2.28	stop	238
6.27	activemq::library::ActiveMQCPP Class Reference	238
6.27.1	Constructor & Destructor Documentation	239
6.27.1.1	ActiveMQCPP	239
6.27.1.2	ActiveMQCPP	239
6.27.1.3	~ActiveMQCPP	239
6.27.2	Member Function Documentation	239
6.27.2.1	initializeLibrary	239
6.27.2.2	initializeLibrary	239

6.27.2.3	operator=	239
6.27.2.4	shutdownLibrary	239
6.28	activemq::commands::ActiveMQDestination Class Reference	240
6.28.1	Constructor & Destructor Documentation	242
6.28.1.1	ActiveMQDestination	242
6.28.1.2	ActiveMQDestination	242
6.28.1.3	~ActiveMQDestination	242
6.28.2	Member Function Documentation	242
6.28.2.1	cloneDataStructure	242
6.28.2.2	copyDataStructure	243
6.28.2.3	createDestination	243
6.28.2.4	createTemporaryName	243
6.28.2.5	equals	243
6.28.2.6	getClientId	244
6.28.2.7	getCMSDestination	244
6.28.2.8	getDataStructureType	244
6.28.2.9	getDestinationType	244
6.28.2.10	getOptions	245
6.28.2.11	getOrderedTarget	245
6.28.2.12	getPhysicalName	245
6.28.2.13	getPhysicalName	245
6.28.2.14	isAdvisory	245
6.28.2.15	isComposite	245
6.28.2.16	isConnectionAdvisory	246
6.28.2.17	isConsumerAdvisory	246
6.28.2.18	isExclusive	246
6.28.2.19	isOrdered	246
6.28.2.20	isProducerAdvisory	246
6.28.2.21	isQueue	246
6.28.2.22	isTemporary	247
6.28.2.23	isTopic	247
6.28.2.24	isWildcard	247
6.28.2.25	setAdvisory	247
6.28.2.26	setExclusive	247
6.28.2.27	setOrdered	247
6.28.2.28	setOrderedTarget	248

6.28.2.29	setPhysicalName	248
6.28.2.30	toString	248
6.28.3	Field Documentation	248
6.28.3.1	advisory	248
6.28.3.2	ADVISORY_PREFIX	248
6.28.3.3	COMPOSITE_SEPARATOR	248
6.28.3.4	CONNECTION_ADVISORY_PREFIX	248
6.28.3.5	CONSUMER_ADVISORY_PREFIX	249
6.28.3.6	DEFAULT_ORDERED_TARGET	249
6.28.3.7	exclusive	249
6.28.3.8	ID_ACTIVEMQDESTINATION	249
6.28.3.9	options	249
6.28.3.10	ordered	249
6.28.3.11	orderedTarget	249
6.28.3.12	physicalName	249
6.28.3.13	PRODUCER_ADVISORY_PREFIX	249
6.28.3.14	QUEUE_QUALIFIED_PREFIX	250
6.28.3.15	TEMP_POSTFIX	250
6.28.3.16	TEMP_PREFIX	250
6.28.3.17	TEMP_QUEUE_QUALIFIED_PREFIX	250
6.28.3.18	TEMP_TOPIC_QUALIFIED_PREFIX	250
6.28.3.19	TOPIC_QUALIFIED_PREFIX	250
6.29	activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller	
	Class Reference	250
6.29.1	Detailed Description	251
6.29.2	Constructor & Destructor Documentation	251
6.29.2.1	ActiveMQDestinationMarshaller	251
6.29.2.2	~ActiveMQDestinationMarshaller	251
6.29.3	Member Function Documentation	251
6.29.3.1	looseMarshal	251
6.29.3.2	looseUnmarshal	252
6.29.3.3	tightMarshal1	252
6.29.3.4	tightMarshal2	253
6.29.3.5	tightUnmarshal	253
6.30	activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller	
	Class Reference	254
6.30.1	Detailed Description	255

6.30.2	Constructor & Destructor Documentation	255
6.30.2.1	ActiveMQDestinationMarshaller	255
6.30.2.2	~ActiveMQDestinationMarshaller	255
6.30.3	Member Function Documentation	255
6.30.3.1	looseMarshal	255
6.30.3.2	looseUnmarshal	256
6.30.3.3	tightMarshal1	256
6.30.3.4	tightMarshal2	257
6.30.3.5	tightUnmarshal	257
6.31	activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller Class Reference	258
6.31.1	Detailed Description	259
6.31.2	Constructor & Destructor Documentation	259
6.31.2.1	ActiveMQDestinationMarshaller	259
6.31.2.2	~ActiveMQDestinationMarshaller	259
6.31.3	Member Function Documentation	259
6.31.3.1	looseMarshal	259
6.31.3.2	looseUnmarshal	259
6.31.3.3	tightMarshal1	260
6.31.3.4	tightMarshal2	260
6.31.3.5	tightUnmarshal	261
6.32	activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller Class Reference	262
6.32.1	Detailed Description	262
6.32.2	Constructor & Destructor Documentation	263
6.32.2.1	ActiveMQDestinationMarshaller	263
6.32.2.2	~ActiveMQDestinationMarshaller	263
6.32.3	Member Function Documentation	263
6.32.3.1	looseMarshal	263
6.32.3.2	looseUnmarshal	263
6.32.3.3	tightMarshal1	264
6.32.3.4	tightMarshal2	264
6.32.3.5	tightUnmarshal	265
6.33	activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller Class Reference	265
6.33.1	Detailed Description	266
6.33.2	Constructor & Destructor Documentation	267

6.33.2.1	ActiveMQDestinationMarshaller	267
6.33.2.2	~ActiveMQDestinationMarshaller	267
6.33.3	Member Function Documentation	267
6.33.3.1	looseMarshal	267
6.33.3.2	looseUnmarshal	267
6.33.3.3	tightMarshal1	268
6.33.3.4	tightMarshal2	268
6.33.3.5	tightUnmarshal	269
6.34	activemq::exceptions::ActiveMQException Class Reference	269
6.34.1	Constructor & Destructor Documentation	270
6.34.1.1	ActiveMQException	270
6.34.1.2	ActiveMQException	270
6.34.1.3	ActiveMQException	270
6.34.1.4	ActiveMQException	271
6.34.1.5	~ActiveMQException	271
6.34.2	Member Function Documentation	271
6.34.2.1	clone	271
6.34.2.2	convertToCMSException	271
6.35	activemq::commands::ActiveMQMapMessage Class Reference	272
6.35.1	Constructor & Destructor Documentation	275
6.35.1.1	ActiveMQMapMessage	275
6.35.1.2	~ActiveMQMapMessage	275
6.35.2	Member Function Documentation	275
6.35.2.1	beforeMarshal	275
6.35.2.2	checkMapIsUnmarshalled	275
6.35.2.3	clearBody	275
6.35.2.4	clone	275
6.35.2.5	cloneDataStructure	276
6.35.2.6	copyDataStructure	276
6.35.2.7	equals	276
6.35.2.8	getBoolean	276
6.35.2.9	getByte	277
6.35.2.10	getBytes	277
6.35.2.11	getChar	277
6.35.2.12	getDataStructureType	278
6.35.2.13	getDouble	278

6.35.2.14	getFloat	278
6.35.2.15	getInt	278
6.35.2.16	getLong	279
6.35.2.17	getMap	279
6.35.2.18	getMap	279
6.35.2.19	getMapNames	279
6.35.2.20	getShort	280
6.35.2.21	getString	280
6.35.2.22	isMarshalAware	280
6.35.2.23	itemExists	280
6.35.2.24	setBoolean	281
6.35.2.25	setByte	281
6.35.2.26	setBytes	281
6.35.2.27	setChar	282
6.35.2.28	setDouble	282
6.35.2.29	setFloat	282
6.35.2.30	setInt	283
6.35.2.31	setLong	283
6.35.2.32	setShort	283
6.35.2.33	setString	284
6.35.2.34	toString	284
6.35.3	Field Documentation	284
6.35.3.1	ID_ACTIVEMQMAPMESSAGE	284
6.36	activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller	
	Class Reference	284
6.36.1	Detailed Description	285
6.36.2	Constructor & Destructor Documentation	286
6.36.2.1	ActiveMQMapMessageMarshaller	286
6.36.2.2	~ActiveMQMapMessageMarshaller	286
6.36.3	Member Function Documentation	286
6.36.3.1	createObject	286
6.36.3.2	getDataStructureType	286
6.36.3.3	looseMarshal	286
6.36.3.4	looseUnmarshal	287
6.36.3.5	tightMarshal1	287
6.36.3.6	tightMarshal2	287
6.36.3.7	tightUnmarshal	288

6.37	activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller	
	Class Reference	288
6.37.1	Detailed Description	289
6.37.2	Constructor & Destructor Documentation	290
	6.37.2.1 ActiveMQMapMessageMarshaller	290
	6.37.2.2 ~ActiveMQMapMessageMarshaller	290
6.37.3	Member Function Documentation	290
	6.37.3.1 createObject	290
	6.37.3.2 getDataStructureType	290
	6.37.3.3 looseMarshal	290
	6.37.3.4 looseUnmarshal	291
	6.37.3.5 tightMarshal1	291
	6.37.3.6 tightMarshal2	291
	6.37.3.7 tightUnmarshal	292
6.38	activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller	
	Class Reference	292
6.38.1	Detailed Description	293
6.38.2	Constructor & Destructor Documentation	294
	6.38.2.1 ActiveMQMapMessageMarshaller	294
	6.38.2.2 ~ActiveMQMapMessageMarshaller	294
6.38.3	Member Function Documentation	294
	6.38.3.1 createObject	294
	6.38.3.2 getDataStructureType	294
	6.38.3.3 looseMarshal	294
	6.38.3.4 looseUnmarshal	295
	6.38.3.5 tightMarshal1	295
	6.38.3.6 tightMarshal2	295
	6.38.3.7 tightUnmarshal	296
6.39	activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller	
	Class Reference	296
6.39.1	Detailed Description	297
6.39.2	Constructor & Destructor Documentation	298
	6.39.2.1 ActiveMQMapMessageMarshaller	298
	6.39.2.2 ~ActiveMQMapMessageMarshaller	298
6.39.3	Member Function Documentation	298
	6.39.3.1 createObject	298
	6.39.3.2 getDataStructureType	298

6.39.3.3	looseMarshal	298
6.39.3.4	looseUnmarshal	299
6.39.3.5	tightMarshal1	299
6.39.3.6	tightMarshal2	299
6.39.3.7	tightUnmarshal	300
6.40	activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller	
	Class Reference	300
6.40.1	Detailed Description	301
6.40.2	Constructor & Destructor Documentation	302
	6.40.2.1 ActiveMQMapMessageMarshaller	302
	6.40.2.2 ~ActiveMQMapMessageMarshaller	302
6.40.3	Member Function Documentation	302
	6.40.3.1 createObject	302
	6.40.3.2 getDataStructureType	302
	6.40.3.3 looseMarshal	302
	6.40.3.4 looseUnmarshal	303
	6.40.3.5 tightMarshal1	303
	6.40.3.6 tightMarshal2	303
	6.40.3.7 tightUnmarshal	304
6.41	activemq::commands::ActiveMQMessage Class Reference	304
	6.41.1 Constructor & Destructor Documentation	305
	6.41.1.1 ActiveMQMessage	305
	6.41.1.2 ~ActiveMQMessage	305
6.41.2	Member Function Documentation	305
	6.41.2.1 clone	305
	6.41.2.2 cloneDataStructure	306
	6.41.2.3 copyDataStructure	306
	6.41.2.4 equals	306
	6.41.2.5 getDataStructureType	306
	6.41.2.6 toString	307
6.41.3	Field Documentation	307
	6.41.3.1 ID_ACTIVEMQMESSAGE	307
6.42	activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller Class	
	Reference	307
6.42.1	Detailed Description	308
6.42.2	Constructor & Destructor Documentation	308
	6.42.2.1 ActiveMQMessageMarshaller	308

6.42.2.2	<code>~ActiveMQMessageMarshaller</code>	308
6.42.3	Member Function Documentation	308
6.42.3.1	<code>createObject</code>	308
6.42.3.2	<code>getDataStructureType</code>	308
6.42.3.3	<code>looseMarshal</code>	309
6.42.3.4	<code>looseUnmarshal</code>	309
6.42.3.5	<code>tightMarshal1</code>	309
6.42.3.6	<code>tightMarshal2</code>	310
6.42.3.7	<code>tightUnmarshal</code>	310
6.43	<code>activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller</code> Class Reference	311
6.43.1	Detailed Description	312
6.43.2	Constructor & Destructor Documentation	312
6.43.2.1	<code>ActiveMQMessageMarshaller</code>	312
6.43.2.2	<code>~ActiveMQMessageMarshaller</code>	312
6.43.3	Member Function Documentation	312
6.43.3.1	<code>createObject</code>	312
6.43.3.2	<code>getDataStructureType</code>	312
6.43.3.3	<code>looseMarshal</code>	313
6.43.3.4	<code>looseUnmarshal</code>	313
6.43.3.5	<code>tightMarshal1</code>	313
6.43.3.6	<code>tightMarshal2</code>	314
6.43.3.7	<code>tightUnmarshal</code>	314
6.44	<code>activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller</code> Class Reference	315
6.44.1	Detailed Description	316
6.44.2	Constructor & Destructor Documentation	316
6.44.2.1	<code>ActiveMQMessageMarshaller</code>	316
6.44.2.2	<code>~ActiveMQMessageMarshaller</code>	316
6.44.3	Member Function Documentation	316
6.44.3.1	<code>createObject</code>	316
6.44.3.2	<code>getDataStructureType</code>	316
6.44.3.3	<code>looseMarshal</code>	317
6.44.3.4	<code>looseUnmarshal</code>	317
6.44.3.5	<code>tightMarshal1</code>	317
6.44.3.6	<code>tightMarshal2</code>	318
6.44.3.7	<code>tightUnmarshal</code>	318

6.45	activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller	Class	
	Reference		319
6.45.1	Detailed Description		320
6.45.2	Constructor & Destructor Documentation		320
6.45.2.1	ActiveMQMessageMarshaller		320
6.45.2.2	~ActiveMQMessageMarshaller		320
6.45.3	Member Function Documentation		320
6.45.3.1	createObject		320
6.45.3.2	getDataStructureType		320
6.45.3.3	looseMarshal		321
6.45.3.4	looseUnmarshal		321
6.45.3.5	tightMarshal1		321
6.45.3.6	tightMarshal2		322
6.45.3.7	tightUnmarshal		322
6.46	activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller	Class	
	Reference		323
6.46.1	Detailed Description		324
6.46.2	Constructor & Destructor Documentation		324
6.46.2.1	ActiveMQMessageMarshaller		324
6.46.2.2	~ActiveMQMessageMarshaller		324
6.46.3	Member Function Documentation		324
6.46.3.1	createObject		324
6.46.3.2	getDataStructureType		324
6.46.3.3	looseMarshal		325
6.46.3.4	looseUnmarshal		325
6.46.3.5	tightMarshal1		325
6.46.3.6	tightMarshal2		326
6.46.3.7	tightUnmarshal		326
6.47	activemq::commands::ActiveMQMessageTemplate< T >	Class Template Reference	327
6.47.1	Constructor & Destructor Documentation		331
6.47.1.1	ActiveMQMessageTemplate		331
6.47.1.2	~ActiveMQMessageTemplate		331
6.47.2	Member Function Documentation		331
6.47.2.1	acknowledge		331
6.47.2.2	clearBody		331
6.47.2.3	clearProperties		331
6.47.2.4	equals		331

6.47.2.5 failIfReadOnlyBody	332
6.47.2.6 failIfReadOnlyProperties	332
6.47.2.7 failIfWriteOnlyBody	332
6.47.2.8 getBooleanProperty	332
6.47.2.9 getByteProperty	332
6.47.2.10 getCMSCorrelationID	333
6.47.2.11 getCMSDeliveryMode	333
6.47.2.12 getCMSDestination	333
6.47.2.13 getCMSExpiration	334
6.47.2.14 getCMSMessageID	334
6.47.2.15 getCMSPriority	334
6.47.2.16 getCMSRedelivered	334
6.47.2.17 getCMSReplyTo	335
6.47.2.18 getCMSTimestamp	335
6.47.2.19 getCMSType	335
6.47.2.20 getDoubleProperty	336
6.47.2.21 getFloatProperty	336
6.47.2.22 getIntProperty	336
6.47.2.23 getLongProperty	337
6.47.2.24 getPropertyNames	337
6.47.2.25 getShortProperty	337
6.47.2.26 getStringProperty	338
6.47.2.27 onSend	338
6.47.2.28 propertyExists	338
6.47.2.29 setBooleanProperty	338
6.47.2.30 setByteProperty	339
6.47.2.31 setCMSCorrelationID	339
6.47.2.32 setCMSDeliveryMode	339
6.47.2.33 setCMSDestination	340
6.47.2.34 setCMSExpiration	340
6.47.2.35 setCMSMessageID	340
6.47.2.36 setCMSPriority	341
6.47.2.37 setCMSRedelivered	341
6.47.2.38 setCMSReplyTo	341
6.47.2.39 setCMSTimestamp	341
6.47.2.40 setCMSType	342

6.47.2.41	setDoubleProperty	342
6.47.2.42	setFloatProperty	342
6.47.2.43	setIntProperty	343
6.47.2.44	setLongProperty	343
6.47.2.45	setShortProperty	343
6.47.2.46	setStringProperty	344
6.48	activemq::commands::ActiveMQObjectMessage Class Reference	344
6.48.1	Constructor & Destructor Documentation	345
6.48.1.1	ActiveMQObjectMessage	345
6.48.1.2	~ActiveMQObjectMessage	345
6.48.2	Member Function Documentation	345
6.48.2.1	clone	345
6.48.2.2	cloneDataStructure	345
6.48.2.3	copyDataStructure	346
6.48.2.4	equals	346
6.48.2.5	getDataStructureType	346
6.48.2.6	toString	346
6.48.3	Field Documentation	347
6.48.3.1	ID_ACTIVEMQOBJECTMESSAGE	347
6.49	activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller Class Reference	347
6.49.1	Detailed Description	348
6.49.2	Constructor & Destructor Documentation	348
6.49.2.1	ActiveMQObjectMessageMarshaller	348
6.49.2.2	~ActiveMQObjectMessageMarshaller	348
6.49.3	Member Function Documentation	348
6.49.3.1	createObject	348
6.49.3.2	getDataStructureType	348
6.49.3.3	looseMarshal	349
6.49.3.4	looseUnmarshal	349
6.49.3.5	tightMarshal1	349
6.49.3.6	tightMarshal2	350
6.49.3.7	tightUnmarshal	350
6.50	activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller Class Reference	351
6.50.1	Detailed Description	352
6.50.2	Constructor & Destructor Documentation	352

6.50.2.1	ActiveMQObjectMessageMarshaller	352
6.50.2.2	~ActiveMQObjectMessageMarshaller	352
6.50.3	Member Function Documentation	352
6.50.3.1	createObject	352
6.50.3.2	getDataStructureType	352
6.50.3.3	looseMarshal	353
6.50.3.4	looseUnmarshal	353
6.50.3.5	tightMarshal1	353
6.50.3.6	tightMarshal2	354
6.50.3.7	tightUnmarshal	354
6.51	activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller	
	Class Reference	355
6.51.1	Detailed Description	356
6.51.2	Constructor & Destructor Documentation	356
6.51.2.1	ActiveMQObjectMessageMarshaller	356
6.51.2.2	~ActiveMQObjectMessageMarshaller	356
6.51.3	Member Function Documentation	356
6.51.3.1	createObject	356
6.51.3.2	getDataStructureType	356
6.51.3.3	looseMarshal	357
6.51.3.4	looseUnmarshal	357
6.51.3.5	tightMarshal1	357
6.51.3.6	tightMarshal2	358
6.51.3.7	tightUnmarshal	358
6.52	activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller	
	Class Reference	359
6.52.1	Detailed Description	360
6.52.2	Constructor & Destructor Documentation	360
6.52.2.1	ActiveMQObjectMessageMarshaller	360
6.52.2.2	~ActiveMQObjectMessageMarshaller	360
6.52.3	Member Function Documentation	360
6.52.3.1	createObject	360
6.52.3.2	getDataStructureType	360
6.52.3.3	looseMarshal	361
6.52.3.4	looseUnmarshal	361
6.52.3.5	tightMarshal1	361
6.52.3.6	tightMarshal2	362

6.52.3.7	tightUnmarshal	362
6.53	activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller Class Reference	363
6.53.1	Detailed Description	364
6.53.2	Constructor & Destructor Documentation	364
6.53.2.1	ActiveMQObjectMessageMarshaller	364
6.53.2.2	~ActiveMQObjectMessageMarshaller	364
6.53.3	Member Function Documentation	364
6.53.3.1	createObject	364
6.53.3.2	getDataStructureType	364
6.53.3.3	looseMarshal	365
6.53.3.4	looseUnmarshal	365
6.53.3.5	tightMarshal1	365
6.53.3.6	tightMarshal2	366
6.53.3.7	tightUnmarshal	366
6.54	activemq::core::ActiveMQProducer Class Reference	367
6.54.1	Constructor & Destructor Documentation	369
6.54.1.1	ActiveMQProducer	369
6.54.1.2	~ActiveMQProducer	369
6.54.2	Member Function Documentation	369
6.54.2.1	close	369
6.54.2.2	getDeliveryMode	369
6.54.2.3	getDisableMessageID	369
6.54.2.4	getDisableMessageTimeStamp	370
6.54.2.5	getPriority	370
6.54.2.6	getProducerId	370
6.54.2.7	getProducerInfo	370
6.54.2.8	getSendTimeout	370
6.54.2.9	getTimeToLive	371
6.54.2.10	isClosed	371
6.54.2.11	onProducerAck	371
6.54.2.12	send	371
6.54.2.13	send	372
6.54.2.14	send	372
6.54.2.15	send	373
6.54.2.16	setDeliveryMode	373
6.54.2.17	setDisableMessageID	373

6.54.2.18	setDisableMessageTimeStamp	374
6.54.2.19	setPriority	374
6.54.2.20	setSendTimeout	374
6.54.2.21	setTimeToLive	374
6.55	activemq::util::ActiveMQProperties Class Reference	374
6.55.1	Detailed Description	376
6.55.2	Constructor & Destructor Documentation	376
6.55.2.1	~ActiveMQProperties	376
6.55.3	Member Function Documentation	376
6.55.3.1	clear	376
6.55.3.2	clone	376
6.55.3.3	copy	376
6.55.3.4	getProperties	377
6.55.3.5	getProperties	377
6.55.3.6	getProperty	377
6.55.3.7	getProperty	377
6.55.3.8	hasProperty	377
6.55.3.9	isEmpty	378
6.55.3.10	remove	378
6.55.3.11	setProperties	378
6.55.3.12	setProperty	378
6.55.3.13	toArray	378
6.55.3.14	toString	379
6.56	activemq::commands::ActiveMQQueue Class Reference	379
6.56.1	Constructor & Destructor Documentation	380
6.56.1.1	ActiveMQQueue	380
6.56.1.2	ActiveMQQueue	380
6.56.1.3	~ActiveMQQueue	380
6.56.2	Member Function Documentation	380
6.56.2.1	clone	380
6.56.2.2	cloneDataStructure	380
6.56.2.3	copy	380
6.56.2.4	copyDataStructure	381
6.56.2.5	equals	381
6.56.2.6	getCMSDestination	381
6.56.2.7	getCMSProperties	381

6.56.2.8	getDataStructureType	382
6.56.2.9	getDestinationType	382
6.56.2.10	getQueueName	382
6.56.2.11	toString	382
6.56.3	Field Documentation	382
6.56.3.1	ID_ACTIVEMQUEUE	382
6.57	activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller Class	
	Reference	382
6.57.1	Detailed Description	383
6.57.2	Constructor & Destructor Documentation	384
6.57.2.1	ActiveMQQueueMarshaller	384
6.57.2.2	~ActiveMQQueueMarshaller	384
6.57.3	Member Function Documentation	384
6.57.3.1	createObject	384
6.57.3.2	getDataStructureType	384
6.57.3.3	looseMarshal	384
6.57.3.4	looseUnmarshal	385
6.57.3.5	tightMarshal1	385
6.57.3.6	tightMarshal2	385
6.57.3.7	tightUnmarshal	386
6.58	activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller Class	
	Reference	386
6.58.1	Detailed Description	387
6.58.2	Constructor & Destructor Documentation	388
6.58.2.1	ActiveMQQueueMarshaller	388
6.58.2.2	~ActiveMQQueueMarshaller	388
6.58.3	Member Function Documentation	388
6.58.3.1	createObject	388
6.58.3.2	getDataStructureType	388
6.58.3.3	looseMarshal	388
6.58.3.4	looseUnmarshal	389
6.58.3.5	tightMarshal1	389
6.58.3.6	tightMarshal2	389
6.58.3.7	tightUnmarshal	390
6.59	activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller Class	
	Reference	390
6.59.1	Detailed Description	391

6.59.2	Constructor & Destructor Documentation	392
6.59.2.1	ActiveMQQueueMarshaller	392
6.59.2.2	~ActiveMQQueueMarshaller	392
6.59.3	Member Function Documentation	392
6.59.3.1	createObject	392
6.59.3.2	getDataStructureType	392
6.59.3.3	looseMarshal	392
6.59.3.4	looseUnmarshal	393
6.59.3.5	tightMarshal1	393
6.59.3.6	tightMarshal2	393
6.59.3.7	tightUnmarshal	394
6.60	activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller Class Reference	394
6.60.1	Detailed Description	395
6.60.2	Constructor & Destructor Documentation	396
6.60.2.1	ActiveMQQueueMarshaller	396
6.60.2.2	~ActiveMQQueueMarshaller	396
6.60.3	Member Function Documentation	396
6.60.3.1	createObject	396
6.60.3.2	getDataStructureType	396
6.60.3.3	looseMarshal	396
6.60.3.4	looseUnmarshal	397
6.60.3.5	tightMarshal1	397
6.60.3.6	tightMarshal2	397
6.60.3.7	tightUnmarshal	398
6.61	activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller Class Reference	398
6.61.1	Detailed Description	399
6.61.2	Constructor & Destructor Documentation	400
6.61.2.1	ActiveMQQueueMarshaller	400
6.61.2.2	~ActiveMQQueueMarshaller	400
6.61.3	Member Function Documentation	400
6.61.3.1	createObject	400
6.61.3.2	getDataStructureType	400
6.61.3.3	looseMarshal	400
6.61.3.4	looseUnmarshal	401
6.61.3.5	tightMarshal1	401

6.61.3.6	tightMarshal2	401
6.61.3.7	tightUnmarshal	402
6.62	activemq::core::ActiveMQSession Class Reference	402
6.62.1	Constructor & Destructor Documentation	406
6.62.1.1	ActiveMQSession	406
6.62.1.2	~ActiveMQSession	406
6.62.2	Member Function Documentation	406
6.62.2.1	acknowledge	406
6.62.2.2	clearMessagesInProgress	407
6.62.2.3	close	407
6.62.2.4	commit	407
6.62.2.5	createBrowser	407
6.62.2.6	createBrowser	407
6.62.2.7	createBytesMessage	408
6.62.2.8	createBytesMessage	408
6.62.2.9	createConsumer	408
6.62.2.10	createConsumer	409
6.62.2.11	createConsumer	409
6.62.2.12	createDurableConsumer	409
6.62.2.13	createMapMessage	410
6.62.2.14	createMessage	410
6.62.2.15	createProducer	410
6.62.2.16	createQueue	411
6.62.2.17	createStreamMessage	411
6.62.2.18	createTemporaryQueue	411
6.62.2.19	createTemporaryTopic	411
6.62.2.20	createTextMessage	412
6.62.2.21	createTextMessage	412
6.62.2.22	createTopic	412
6.62.2.23	deliverAcks	412
6.62.2.24	dispatch	413
6.62.2.25	disposeOf	413
6.62.2.26	disposeOf	413
6.62.2.27	doStartTransaction	413
6.62.2.28	fire	414
6.62.2.29	getAcknowledgeMode	414

6.62.2.30	getConnection	414
6.62.2.31	getExceptionListener	414
6.62.2.32	getLastDeliveredSequenceId	414
6.62.2.33	getSessionId	414
6.62.2.34	getSessionInfo	415
6.62.2.35	isAutoAcknowledge	415
6.62.2.36	isClientAcknowledge	415
6.62.2.37	isDupsOkAcknowledge	415
6.62.2.38	isIndividualAcknowledge	415
6.62.2.39	isStarted	415
6.62.2.40	isTransacted	415
6.62.2.41	oneway	415
6.62.2.42	recover	416
6.62.2.43	redispach	416
6.62.2.44	rollback	416
6.62.2.45	send	416
6.62.2.46	setLastDeliveredSequenceId	417
6.62.2.47	start	417
6.62.2.48	stop	417
6.62.2.49	syncRequest	417
6.62.2.50	unsubscribe	418
6.62.2.51	wakeup	418
6.62.3	Friends And Related Function Documentation	418
6.62.3.1	ActiveMQSessionExecutor	418
6.63	activemq::core::ActiveMQSessionExecutor Class Reference	418
6.63.1	Detailed Description	419
6.63.2	Constructor & Destructor Documentation	419
6.63.2.1	ActiveMQSessionExecutor	419
6.63.2.2	~ActiveMQSessionExecutor	419
6.63.3	Member Function Documentation	420
6.63.3.1	clear	420
6.63.3.2	clearMessagesInProgress	420
6.63.3.3	close	420
6.63.3.4	execute	420
6.63.3.5	executeFirst	420
6.63.3.6	getUnconsumedMessages	420

6.63.3.7	hasUnconsumedMessages	421
6.63.3.8	isEmpty	421
6.63.3.9	isRunning	421
6.63.3.10	iterate	421
6.63.3.11	start	421
6.63.3.12	stop	421
6.63.3.13	wakeup	421
6.64	activemq::commands::ActiveMQStreamMessage Class Reference	422
6.64.1	Constructor & Destructor Documentation	425
6.64.1.1	ActiveMQStreamMessage	425
6.64.1.2	~ActiveMQStreamMessage	425
6.64.2	Member Function Documentation	425
6.64.2.1	clearBody	425
6.64.2.2	clone	425
6.64.2.3	cloneDataStructure	425
6.64.2.4	copyDataStructure	425
6.64.2.5	equals	426
6.64.2.6	getDataStructureType	426
6.64.2.7	onSend	426
6.64.2.8	readBoolean	426
6.64.2.9	readByte	427
6.64.2.10	readBytes	427
6.64.2.11	readBytes	428
6.64.2.12	readChar	428
6.64.2.13	readDouble	429
6.64.2.14	readFloat	429
6.64.2.15	readInt	429
6.64.2.16	readLong	430
6.64.2.17	readShort	430
6.64.2.18	readString	431
6.64.2.19	readUnsignedShort	431
6.64.2.20	reset	431
6.64.2.21	toString	432
6.64.2.22	writeBoolean	432
6.64.2.23	writeByte	432
6.64.2.24	writeBytes	432

6.64.2.25	writeBytes	433
6.64.2.26	writeChar	433
6.64.2.27	writeDouble	433
6.64.2.28	writeFloat	434
6.64.2.29	writeInt	434
6.64.2.30	writeLong	434
6.64.2.31	writeShort	435
6.64.2.32	writeString	435
6.64.2.33	writeUnsignedShort	435
6.64.3	Field Documentation	435
6.64.3.1	ID_ACTIVEMQSTREAMMESSAGE	435
6.65	activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller	
	Class Reference	436
6.65.1	Detailed Description	437
6.65.2	Constructor & Destructor Documentation	437
6.65.2.1	ActiveMQStreamMessageMarshaller	437
6.65.2.2	~ActiveMQStreamMessageMarshaller	437
6.65.3	Member Function Documentation	437
6.65.3.1	createObject	437
6.65.3.2	getDataStructureType	437
6.65.3.3	looseMarshal	437
6.65.3.4	looseUnmarshal	438
6.65.3.5	tightMarshal1	438
6.65.3.6	tightMarshal2	439
6.65.3.7	tightUnmarshal	439
6.66	activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller	
	Class Reference	439
6.66.1	Detailed Description	440
6.66.2	Constructor & Destructor Documentation	441
6.66.2.1	ActiveMQStreamMessageMarshaller	441
6.66.2.2	~ActiveMQStreamMessageMarshaller	441
6.66.3	Member Function Documentation	441
6.66.3.1	createObject	441
6.66.3.2	getDataStructureType	441
6.66.3.3	looseMarshal	441
6.66.3.4	looseUnmarshal	442
6.66.3.5	tightMarshal1	442

6.66.3.6	tightMarshal2	442
6.66.3.7	tightUnmarshal	443
6.67	activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller	
	Class Reference	443
6.67.1	Detailed Description	444
6.67.2	Constructor & Destructor Documentation	445
6.67.2.1	ActiveMQStreamMessageMarshaller	445
6.67.2.2	~ActiveMQStreamMessageMarshaller	445
6.67.3	Member Function Documentation	445
6.67.3.1	createObject	445
6.67.3.2	getDataStructureType	445
6.67.3.3	looseMarshal	445
6.67.3.4	looseUnmarshal	446
6.67.3.5	tightMarshal1	446
6.67.3.6	tightMarshal2	446
6.67.3.7	tightUnmarshal	447
6.68	activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller	
	Class Reference	447
6.68.1	Detailed Description	448
6.68.2	Constructor & Destructor Documentation	449
6.68.2.1	ActiveMQStreamMessageMarshaller	449
6.68.2.2	~ActiveMQStreamMessageMarshaller	449
6.68.3	Member Function Documentation	449
6.68.3.1	createObject	449
6.68.3.2	getDataStructureType	449
6.68.3.3	looseMarshal	449
6.68.3.4	looseUnmarshal	450
6.68.3.5	tightMarshal1	450
6.68.3.6	tightMarshal2	450
6.68.3.7	tightUnmarshal	451
6.69	activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller	
	Class Reference	451
6.69.1	Detailed Description	452
6.69.2	Constructor & Destructor Documentation	453
6.69.2.1	ActiveMQStreamMessageMarshaller	453
6.69.2.2	~ActiveMQStreamMessageMarshaller	453
6.69.3	Member Function Documentation	453

6.69.3.1	createObject	453
6.69.3.2	getDataStructureType	453
6.69.3.3	looseMarshal	453
6.69.3.4	looseUnmarshal	454
6.69.3.5	tightMarshal1	454
6.69.3.6	tightMarshal2	454
6.69.3.7	tightUnmarshal	455
6.70	activemq::commands::ActiveMQTempDestination Class Reference	455
6.70.1	Constructor & Destructor Documentation	457
6.70.1.1	ActiveMQTempDestination	457
6.70.1.2	ActiveMQTempDestination	457
6.70.1.3	~ActiveMQTempDestination	457
6.70.2	Member Function Documentation	457
6.70.2.1	cloneDataStructure	457
6.70.2.2	close	457
6.70.2.3	copyDataStructure	457
6.70.2.4	equals	457
6.70.2.5	getDataStructureType	458
6.70.2.6	setConnection	458
6.70.2.7	toString	458
6.70.3	Field Documentation	458
6.70.3.1	connection	458
6.70.3.2	ID_ACTIVEMQTEMPDESTINATION	458
6.71	activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller Class Reference	459
6.71.1	Detailed Description	459
6.71.2	Constructor & Destructor Documentation	460
6.71.2.1	ActiveMQTempDestinationMarshaller	460
6.71.2.2	~ActiveMQTempDestinationMarshaller	460
6.71.3	Member Function Documentation	460
6.71.3.1	looseMarshal	460
6.71.3.2	looseUnmarshal	460
6.71.3.3	tightMarshal1	461
6.71.3.4	tightMarshal2	461
6.71.3.5	tightUnmarshal	462
6.72	activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller Class Reference	462

6.72.1	Detailed Description	463
6.72.2	Constructor & Destructor Documentation	463
6.72.2.1	ActiveMQTempDestinationMarshaller	463
6.72.2.2	~ActiveMQTempDestinationMarshaller	463
6.72.3	Member Function Documentation	463
6.72.3.1	looseMarshal	463
6.72.3.2	looseUnmarshal	464
6.72.3.3	tightMarshal1	464
6.72.3.4	tightMarshal2	465
6.72.3.5	tightUnmarshal	465
6.73	activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller Class Reference	466
6.73.1	Detailed Description	467
6.73.2	Constructor & Destructor Documentation	467
6.73.2.1	ActiveMQTempDestinationMarshaller	467
6.73.2.2	~ActiveMQTempDestinationMarshaller	467
6.73.3	Member Function Documentation	467
6.73.3.1	looseMarshal	467
6.73.3.2	looseUnmarshal	468
6.73.3.3	tightMarshal1	468
6.73.3.4	tightMarshal2	469
6.73.3.5	tightUnmarshal	469
6.74	activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller Class Reference	470
6.74.1	Detailed Description	470
6.74.2	Constructor & Destructor Documentation	471
6.74.2.1	ActiveMQTempDestinationMarshaller	471
6.74.2.2	~ActiveMQTempDestinationMarshaller	471
6.74.3	Member Function Documentation	471
6.74.3.1	looseMarshal	471
6.74.3.2	looseUnmarshal	471
6.74.3.3	tightMarshal1	472
6.74.3.4	tightMarshal2	472
6.74.3.5	tightUnmarshal	473
6.75	activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller Class Reference	473
6.75.1	Detailed Description	474

6.75.2	Constructor & Destructor Documentation	474
6.75.2.1	ActiveMQTempDestinationMarshaller	474
6.75.2.2	~ActiveMQTempDestinationMarshaller	474
6.75.3	Member Function Documentation	474
6.75.3.1	looseMarshal	474
6.75.3.2	looseUnmarshal	475
6.75.3.3	tightMarshal1	475
6.75.3.4	tightMarshal2	476
6.75.3.5	tightUnmarshal	476
6.76	activemq::commands::ActiveMQTempQueue Class Reference	477
6.76.1	Constructor & Destructor Documentation	478
6.76.1.1	ActiveMQTempQueue	478
6.76.1.2	ActiveMQTempQueue	478
6.76.1.3	~ActiveMQTempQueue	478
6.76.2	Member Function Documentation	478
6.76.2.1	clone	478
6.76.2.2	cloneDataStructure	478
6.76.2.3	copy	479
6.76.2.4	copyDataStructure	479
6.76.2.5	destroy	479
6.76.2.6	equals	479
6.76.2.7	getCMSDestination	480
6.76.2.8	getCMSProperties	480
6.76.2.9	getDataStructureType	480
6.76.2.10	getDestinationType	480
6.76.2.11	getQueueName	480
6.76.2.12	toString	481
6.76.3	Field Documentation	481
6.76.3.1	ID_ACTIVEMQTEMPQUEUE	481
6.77	activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller Class Reference	481
6.77.1	Detailed Description	482
6.77.2	Constructor & Destructor Documentation	482
6.77.2.1	ActiveMQTempQueueMarshaller	482
6.77.2.2	~ActiveMQTempQueueMarshaller	482
6.77.3	Member Function Documentation	482
6.77.3.1	createObject	482

6.77.3.2	getDataStructureType	482
6.77.3.3	looseMarshal	483
6.77.3.4	looseUnmarshal	483
6.77.3.5	tightMarshal1	483
6.77.3.6	tightMarshal2	484
6.77.3.7	tightUnmarshal	484
6.78	activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller	
	Class Reference	485
6.78.1	Detailed Description	486
6.78.2	Constructor & Destructor Documentation	486
6.78.2.1	ActiveMQTempQueueMarshaller	486
6.78.2.2	~ActiveMQTempQueueMarshaller	486
6.78.3	Member Function Documentation	486
6.78.3.1	createObject	486
6.78.3.2	getDataStructureType	486
6.78.3.3	looseMarshal	487
6.78.3.4	looseUnmarshal	487
6.78.3.5	tightMarshal1	487
6.78.3.6	tightMarshal2	488
6.78.3.7	tightUnmarshal	488
6.79	activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller	
	Class Reference	489
6.79.1	Detailed Description	490
6.79.2	Constructor & Destructor Documentation	490
6.79.2.1	ActiveMQTempQueueMarshaller	490
6.79.2.2	~ActiveMQTempQueueMarshaller	490
6.79.3	Member Function Documentation	490
6.79.3.1	createObject	490
6.79.3.2	getDataStructureType	490
6.79.3.3	looseMarshal	491
6.79.3.4	looseUnmarshal	491
6.79.3.5	tightMarshal1	491
6.79.3.6	tightMarshal2	492
6.79.3.7	tightUnmarshal	492
6.80	activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller	
	Class Reference	493
6.80.1	Detailed Description	494

6.80.2	Constructor & Destructor Documentation	494
6.80.2.1	ActiveMQTempQueueMarshaller	494
6.80.2.2	~ActiveMQTempQueueMarshaller	494
6.80.3	Member Function Documentation	494
6.80.3.1	createObject	494
6.80.3.2	getDataStructureType	494
6.80.3.3	looseMarshal	495
6.80.3.4	looseUnmarshal	495
6.80.3.5	tightMarshal1	495
6.80.3.6	tightMarshal2	496
6.80.3.7	tightUnmarshal	496
6.81	activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller Class Reference	497
6.81.1	Detailed Description	498
6.81.2	Constructor & Destructor Documentation	498
6.81.2.1	ActiveMQTempQueueMarshaller	498
6.81.2.2	~ActiveMQTempQueueMarshaller	498
6.81.3	Member Function Documentation	498
6.81.3.1	createObject	498
6.81.3.2	getDataStructureType	498
6.81.3.3	looseMarshal	499
6.81.3.4	looseUnmarshal	499
6.81.3.5	tightMarshal1	499
6.81.3.6	tightMarshal2	500
6.81.3.7	tightUnmarshal	500
6.82	activemq::commands::ActiveMQTempTopic Class Reference	501
6.82.1	Constructor & Destructor Documentation	502
6.82.1.1	ActiveMQTempTopic	502
6.82.1.2	ActiveMQTempTopic	502
6.82.1.3	~ActiveMQTempTopic	502
6.82.2	Member Function Documentation	502
6.82.2.1	clone	502
6.82.2.2	cloneDataStructure	502
6.82.2.3	copy	503
6.82.2.4	copyDataStructure	503
6.82.2.5	destroy	503
6.82.2.6	equals	503

6.82.2.7	getCMSDestination	503
6.82.2.8	getCMSProperties	504
6.82.2.9	getDataStructureType	504
6.82.2.10	getDestinationType	504
6.82.2.11	getTopicName	504
6.82.2.12	toString	504
6.82.3	Field Documentation	505
6.82.3.1	ID_ACTIVEMQTEMPTOPIC	505
6.83	activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller	
	Class Reference	505
6.83.1	Detailed Description	506
6.83.2	Constructor & Destructor Documentation	506
6.83.2.1	ActiveMQTempTopicMarshaller	506
6.83.2.2	~ActiveMQTempTopicMarshaller	506
6.83.3	Member Function Documentation	506
6.83.3.1	createObject	506
6.83.3.2	getDataStructureType	506
6.83.3.3	looseMarshal	507
6.83.3.4	looseUnmarshal	507
6.83.3.5	tightMarshal1	507
6.83.3.6	tightMarshal2	508
6.83.3.7	tightUnmarshal	508
6.84	activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller	
	Class Reference	509
6.84.1	Detailed Description	510
6.84.2	Constructor & Destructor Documentation	510
6.84.2.1	ActiveMQTempTopicMarshaller	510
6.84.2.2	~ActiveMQTempTopicMarshaller	510
6.84.3	Member Function Documentation	510
6.84.3.1	createObject	510
6.84.3.2	getDataStructureType	510
6.84.3.3	looseMarshal	511
6.84.3.4	looseUnmarshal	511
6.84.3.5	tightMarshal1	511
6.84.3.6	tightMarshal2	512
6.84.3.7	tightUnmarshal	512

6.85	activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller	
	Class Reference	513
6.85.1	Detailed Description	514
6.85.2	Constructor & Destructor Documentation	514
	6.85.2.1 ActiveMQTempTopicMarshaller	514
	6.85.2.2 ~ActiveMQTempTopicMarshaller	514
6.85.3	Member Function Documentation	514
	6.85.3.1 createObject	514
	6.85.3.2 getDataStructureType	514
	6.85.3.3 looseMarshal	515
	6.85.3.4 looseUnmarshal	515
	6.85.3.5 tightMarshal1	515
	6.85.3.6 tightMarshal2	516
	6.85.3.7 tightUnmarshal	516
6.86	activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller	
	Class Reference	517
6.86.1	Detailed Description	518
6.86.2	Constructor & Destructor Documentation	518
	6.86.2.1 ActiveMQTempTopicMarshaller	518
	6.86.2.2 ~ActiveMQTempTopicMarshaller	518
6.86.3	Member Function Documentation	518
	6.86.3.1 createObject	518
	6.86.3.2 getDataStructureType	518
	6.86.3.3 looseMarshal	519
	6.86.3.4 looseUnmarshal	519
	6.86.3.5 tightMarshal1	519
	6.86.3.6 tightMarshal2	520
	6.86.3.7 tightUnmarshal	520
6.87	activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller	
	Class Reference	521
6.87.1	Detailed Description	522
6.87.2	Constructor & Destructor Documentation	522
	6.87.2.1 ActiveMQTempTopicMarshaller	522
	6.87.2.2 ~ActiveMQTempTopicMarshaller	522
6.87.3	Member Function Documentation	522
	6.87.3.1 createObject	522
	6.87.3.2 getDataStructureType	522

6.87.3.3	looseMarshal	523
6.87.3.4	looseUnmarshal	523
6.87.3.5	tightMarshal1	523
6.87.3.6	tightMarshal2	524
6.87.3.7	tightUnmarshal	524
6.88	activemq::commands::ActiveMQTextMessage Class Reference	525
6.88.1	Constructor & Destructor Documentation	526
6.88.1.1	ActiveMQTextMessage	526
6.88.1.2	~ActiveMQTextMessage	526
6.88.2	Member Function Documentation	526
6.88.2.1	beforeMarshal	526
6.88.2.2	clearBody	527
6.88.2.3	clone	527
6.88.2.4	cloneDataStructure	527
6.88.2.5	copyDataStructure	527
6.88.2.6	equals	527
6.88.2.7	getDataStructureType	528
6.88.2.8	getSize	528
6.88.2.9	getText	528
6.88.2.10	setText	528
6.88.2.11	setText	529
6.88.2.12	toString	529
6.88.3	Field Documentation	529
6.88.3.1	ID_ACTIVEMQTEXTMESSAGE	529
6.88.3.2	text	529
6.89	activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller Class Reference	529
6.89.1	Detailed Description	530
6.89.2	Constructor & Destructor Documentation	531
6.89.2.1	ActiveMQTextMessageMarshaller	531
6.89.2.2	~ActiveMQTextMessageMarshaller	531
6.89.3	Member Function Documentation	531
6.89.3.1	createObject	531
6.89.3.2	getDataStructureType	531
6.89.3.3	looseMarshal	531
6.89.3.4	looseUnmarshal	532
6.89.3.5	tightMarshal1	532

6.89.3.6	tightMarshal2	532
6.89.3.7	tightUnmarshal	533
6.90	activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller	
	Class Reference	533
6.90.1	Detailed Description	534
6.90.2	Constructor & Destructor Documentation	535
6.90.2.1	ActiveMQTextMessageMarshaller	535
6.90.2.2	~ActiveMQTextMessageMarshaller	535
6.90.3	Member Function Documentation	535
6.90.3.1	createObject	535
6.90.3.2	getDataStructureType	535
6.90.3.3	looseMarshal	535
6.90.3.4	looseUnmarshal	536
6.90.3.5	tightMarshal1	536
6.90.3.6	tightMarshal2	536
6.90.3.7	tightUnmarshal	537
6.91	activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller	
	Class Reference	537
6.91.1	Detailed Description	538
6.91.2	Constructor & Destructor Documentation	539
6.91.2.1	ActiveMQTextMessageMarshaller	539
6.91.2.2	~ActiveMQTextMessageMarshaller	539
6.91.3	Member Function Documentation	539
6.91.3.1	createObject	539
6.91.3.2	getDataStructureType	539
6.91.3.3	looseMarshal	539
6.91.3.4	looseUnmarshal	540
6.91.3.5	tightMarshal1	540
6.91.3.6	tightMarshal2	540
6.91.3.7	tightUnmarshal	541
6.92	activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller	
	Class Reference	541
6.92.1	Detailed Description	542
6.92.2	Constructor & Destructor Documentation	543
6.92.2.1	ActiveMQTextMessageMarshaller	543
6.92.2.2	~ActiveMQTextMessageMarshaller	543
6.92.3	Member Function Documentation	543

6.92.3.1	createObject	543
6.92.3.2	getDataStructureType	543
6.92.3.3	looseMarshal	543
6.92.3.4	looseUnmarshal	544
6.92.3.5	tightMarshal1	544
6.92.3.6	tightMarshal2	544
6.92.3.7	tightUnmarshal	545
6.93	activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller Class Reference	545
6.93.1	Detailed Description	546
6.93.2	Constructor & Destructor Documentation	547
6.93.2.1	ActiveMQTextMessageMarshaller	547
6.93.2.2	~ActiveMQTextMessageMarshaller	547
6.93.3	Member Function Documentation	547
6.93.3.1	createObject	547
6.93.3.2	getDataStructureType	547
6.93.3.3	looseMarshal	547
6.93.3.4	looseUnmarshal	548
6.93.3.5	tightMarshal1	548
6.93.3.6	tightMarshal2	548
6.93.3.7	tightUnmarshal	549
6.94	activemq::commands::ActiveMQTopic Class Reference	549
6.94.1	Constructor & Destructor Documentation	551
6.94.1.1	ActiveMQTopic	551
6.94.1.2	ActiveMQTopic	551
6.94.1.3	~ActiveMQTopic	551
6.94.2	Member Function Documentation	551
6.94.2.1	clone	551
6.94.2.2	cloneDataStructure	551
6.94.2.3	copy	551
6.94.2.4	copyDataStructure	551
6.94.2.5	equals	552
6.94.2.6	getCMSDestination	552
6.94.2.7	getCMSProperties	552
6.94.2.8	getDataStructureType	552
6.94.2.9	getDestinationType	552
6.94.2.10	getTopicName	553

6.94.2.11	toString	553
6.94.3	Field Documentation	553
6.94.3.1	ID_ACTIVEMQTOPIC	553
6.95	activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller Class Reference	553
6.95.1	Detailed Description	554
6.95.2	Constructor & Destructor Documentation	554
6.95.2.1	ActiveMQTopicMarshaller	554
6.95.2.2	~ActiveMQTopicMarshaller	554
6.95.3	Member Function Documentation	554
6.95.3.1	createObject	554
6.95.3.2	getDataStructureType	555
6.95.3.3	looseMarshal	555
6.95.3.4	looseUnmarshal	555
6.95.3.5	tightMarshal1	556
6.95.3.6	tightMarshal2	556
6.95.3.7	tightUnmarshal	557
6.96	activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller Class Reference	557
6.96.1	Detailed Description	558
6.96.2	Constructor & Destructor Documentation	558
6.96.2.1	ActiveMQTopicMarshaller	558
6.96.2.2	~ActiveMQTopicMarshaller	558
6.96.3	Member Function Documentation	558
6.96.3.1	createObject	558
6.96.3.2	getDataStructureType	559
6.96.3.3	looseMarshal	559
6.96.3.4	looseUnmarshal	559
6.96.3.5	tightMarshal1	560
6.96.3.6	tightMarshal2	560
6.96.3.7	tightUnmarshal	560
6.97	activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller Class Reference	561
6.97.1	Detailed Description	562
6.97.2	Constructor & Destructor Documentation	562
6.97.2.1	ActiveMQTopicMarshaller	562
6.97.2.2	~ActiveMQTopicMarshaller	562

6.97.3	Member Function Documentation	562
6.97.3.1	createObject	562
6.97.3.2	getDataStructureType	562
6.97.3.3	looseMarshal	563
6.97.3.4	looseUnmarshal	563
6.97.3.5	tightMarshal1	563
6.97.3.6	tightMarshal2	564
6.97.3.7	tightUnmarshal	564
6.98	activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller Class Reference	565
6.98.1	Detailed Description	566
6.98.2	Constructor & Destructor Documentation	566
6.98.2.1	ActiveMQTopicMarshaller	566
6.98.2.2	~ActiveMQTopicMarshaller	566
6.98.3	Member Function Documentation	566
6.98.3.1	createObject	566
6.98.3.2	getDataStructureType	566
6.98.3.3	looseMarshal	567
6.98.3.4	looseUnmarshal	567
6.98.3.5	tightMarshal1	567
6.98.3.6	tightMarshal2	568
6.98.3.7	tightUnmarshal	568
6.99	activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller Class Reference	569
6.99.1	Detailed Description	570
6.99.2	Constructor & Destructor Documentation	570
6.99.2.1	ActiveMQTopicMarshaller	570
6.99.2.2	~ActiveMQTopicMarshaller	570
6.99.3	Member Function Documentation	570
6.99.3.1	createObject	570
6.99.3.2	getDataStructureType	570
6.99.3.3	looseMarshal	571
6.99.3.4	looseUnmarshal	571
6.99.3.5	tightMarshal1	571
6.99.3.6	tightMarshal2	572
6.99.3.7	tightUnmarshal	572
6.100	activemq::core::ActiveMQTransactionContext Class Reference	573

6.100.1 Detailed Description	574
6.100.2 Constructor & Destructor Documentation	574
6.100.2.1 ActiveMQTransactionContext	574
6.100.2.2 ~ActiveMQTransactionContext	574
6.100.3 Member Function Documentation	574
6.100.3.1 addSynchronization	574
6.100.3.2 begin	574
6.100.3.3 commit	575
6.100.3.4 getMaximumRedeliveries	575
6.100.3.5 getRedeliveryDelay	575
6.100.3.6 getTransactionId	575
6.100.3.7 isInTransaction	575
6.100.3.8 removeSynchronization	576
6.100.3.9 rollback	576
6.101decaf::lang::Appendable Class Reference	576
6.101.1 Constructor & Destructor Documentation	577
6.101.1.1 ~Appendable	577
6.101.2 Member Function Documentation	577
6.101.2.1 append	577
6.101.2.2 append	577
6.101.2.3 append	577
6.102decaf::internal::AprPool Class Reference	578
6.102.1 Detailed Description	578
6.102.2 Constructor & Destructor Documentation	579
6.102.2.1 AprPool	579
6.102.2.2 ~AprPool	579
6.102.3 Member Function Documentation	579
6.102.3.1 cleanup	579
6.102.3.2 getAprPool	579
6.102.3.3 getGlobalPool	579
6.103decaf::util::concurrent::atomic::AtomicBoolean Class Reference	579
6.103.1 Detailed Description	580
6.103.2 Constructor & Destructor Documentation	580
6.103.2.1 AtomicBoolean	580
6.103.2.2 AtomicBoolean	580
6.103.2.3 ~AtomicBoolean	580

6.103.3 Member Function Documentation	580
6.103.3.1 compareAndSet	580
6.103.3.2 get	581
6.103.3.3 getAndSet	581
6.103.3.4 set	581
6.103.3.5 toString	581
6.104 decaf::util::concurrent::atomic::AtomicInteger Class Reference	582
6.104.1 Detailed Description	583
6.104.2 Constructor & Destructor Documentation	583
6.104.2.1 AtomicInteger	583
6.104.2.2 AtomicInteger	583
6.104.2.3 ~AtomicInteger	583
6.104.3 Member Function Documentation	583
6.104.3.1 addAndGet	583
6.104.3.2 compareAndSet	584
6.104.3.3 decrementAndGet	584
6.104.3.4 doubleValue	584
6.104.3.5 floatValue	584
6.104.3.6 get	585
6.104.3.7 getAndAdd	585
6.104.3.8 getAndDecrement	585
6.104.3.9 getAndIncrement	585
6.104.3.10 getAndSet	585
6.104.3.11 incrementAndGet	586
6.104.3.12 intValue	586
6.104.3.13 longValue	586
6.104.3.14 set	586
6.104.3.15 toString	587
6.105 decaf::lang::AtomicRefCounter Class Reference	587
6.105.1 Constructor & Destructor Documentation	588
6.105.1.1 AtomicRefCounter	588
6.105.1.2 AtomicRefCounter	588
6.105.2 Member Function Documentation	588
6.105.2.1 release	588
6.105.2.2 swap	589
6.106 decaf::util::concurrent::atomic::AtomicReference< T > Class Template Reference	589

6.106.1 Detailed Description	590
6.106.2 Constructor & Destructor Documentation	590
6.106.2.1 AtomicReference	590
6.106.2.2 AtomicReference	590
6.106.2.3 ~AtomicReference	590
6.106.3 Member Function Documentation	590
6.106.3.1 compareAndSet	590
6.106.3.2 get	590
6.106.3.3 getAndSet	591
6.106.3.4 set	591
6.106.3.5 toString	591
6.107activemq::transport::failover::BackupTransport Class Reference	591
6.107.1 Constructor & Destructor Documentation	592
6.107.1.1 BackupTransport	592
6.107.1.2 ~BackupTransport	592
6.107.2 Member Function Documentation	592
6.107.2.1 getTransport	592
6.107.2.2 getUri	593
6.107.2.3 isClosed	593
6.107.2.4 onException	593
6.107.2.5 setClosed	593
6.107.2.6 setTransport	593
6.107.2.7 setUri	594
6.108activemq::transport::failover::BackupTransportPool Class Reference	594
6.108.1 Constructor & Destructor Documentation	595
6.108.1.1 BackupTransportPool	595
6.108.1.2 BackupTransportPool	595
6.108.1.3 ~BackupTransportPool	595
6.108.2 Member Function Documentation	595
6.108.2.1 getBackup	595
6.108.2.2 getBackupPoolSize	595
6.108.2.3 isEnabled	595
6.108.2.4 isPending	596
6.108.2.5 iterate	596
6.108.2.6 setBackupPoolSize	596
6.108.2.7 setEnabled	596

6.108.3 Friends And Related Function Documentation	596
6.108.3.1 BackupTransport	596
6.109 activemq::commands::BaseCommand Class Reference	596
6.109.1 Constructor & Destructor Documentation	598
6.109.1.1 BaseCommand	598
6.109.1.2 ~BaseCommand	598
6.109.2 Member Function Documentation	598
6.109.2.1 copyDataStructure	598
6.109.2.2 equals	598
6.109.2.3 getCommandId	599
6.109.2.4 isBrokerInfo	600
6.109.2.5 isConnectionInfo	600
6.109.2.6 isConsumerInfo	600
6.109.2.7 isKeepAliveInfo	600
6.109.2.8 isMessage	600
6.109.2.9 isMessageAck	600
6.109.2.10 isMessageDispatch	600
6.109.2.11 isMessageDispatchNotification	601
6.109.2.12 isProducerAck	601
6.109.2.13 isProducerInfo	601
6.109.2.14 isRemoveInfo	601
6.109.2.15 isRemoveSubscriptionInfo	601
6.109.2.16 isResponse	601
6.109.2.17 isResponseRequired	601
6.109.2.18 isShutdownInfo	602
6.109.2.19 isTransactionInfo	602
6.109.2.20 isWireFormatInfo	602
6.109.2.21 setCommandId	602
6.109.2.22 setResponseRequired	602
6.109.2.23 toString	602
6.110 activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller Class Reference	603
6.110.1 Detailed Description	604
6.110.2 Constructor & Destructor Documentation	604
6.110.2.1 BaseCommandMarshaller	604
6.110.2.2 ~BaseCommandMarshaller	604
6.110.3 Member Function Documentation	604

6.110.3.1 looseMarshal	604
6.110.3.2 looseUnmarshal	605
6.110.3.3 tightMarshal1	606
6.110.3.4 tightMarshal2	608
6.110.3.5 tightUnmarshal	609
6.111activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller Class Reference	610
6.111.1 Detailed Description	611
6.111.2 Constructor & Destructor Documentation	611
6.111.2.1 BaseCommandMarshaller	611
6.111.2.2 ~BaseCommandMarshaller	611
6.111.3 Member Function Documentation	611
6.111.3.1 looseMarshal	611
6.111.3.2 looseUnmarshal	612
6.111.3.3 tightMarshal1	613
6.111.3.4 tightMarshal2	614
6.111.3.5 tightUnmarshal	615
6.112activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller Class Reference	616
6.112.1 Detailed Description	617
6.112.2 Constructor & Destructor Documentation	618
6.112.2.1 BaseCommandMarshaller	618
6.112.2.2 ~BaseCommandMarshaller	618
6.112.3 Member Function Documentation	618
6.112.3.1 looseMarshal	618
6.112.3.2 looseUnmarshal	619
6.112.3.3 tightMarshal1	620
6.112.3.4 tightMarshal2	621
6.112.3.5 tightUnmarshal	622
6.113activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller Class Reference	623
6.113.1 Detailed Description	624
6.113.2 Constructor & Destructor Documentation	624
6.113.2.1 BaseCommandMarshaller	624
6.113.2.2 ~BaseCommandMarshaller	624
6.113.3 Member Function Documentation	624
6.113.3.1 looseMarshal	624

6.113.3.2 looseUnmarshal	625
6.113.3.3 tightMarshal1	626
6.113.3.4 tightMarshal2	628
6.113.3.5 tightUnmarshal	629
6.114activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller Class Reference	630
6.114.1 Detailed Description	631
6.114.2 Constructor & Destructor Documentation	631
6.114.2.1 BaseCommandMarshaller	631
6.114.2.2 ~BaseCommandMarshaller	631
6.114.3 Member Function Documentation	631
6.114.3.1 looseMarshal	631
6.114.3.2 looseUnmarshal	632
6.114.3.3 tightMarshal1	633
6.114.3.4 tightMarshal2	634
6.114.3.5 tightUnmarshal	635
6.115activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller Class Reference	636
6.115.1 Detailed Description	642
6.115.2 Constructor & Destructor Documentation	643
6.115.2.1 ~BaseDataStreamMarshaller	643
6.115.3 Member Function Documentation	643
6.115.3.1 looseMarshal	643
6.115.3.2 looseMarshalBrokerError	643
6.115.3.3 looseMarshalCachedObject	643
6.115.3.4 looseMarshalLong	644
6.115.3.5 looseMarshalNestedObject	644
6.115.3.6 looseMarshalObjectArray	645
6.115.3.7 looseMarshalString	645
6.115.3.8 looseUnmarshal	645
6.115.3.9 looseUnmarshalBrokerError	646
6.115.3.10looseUnmarshalByteArray	646
6.115.3.11looseUnmarshalCachedObject	646
6.115.3.12looseUnmarshalConstByteArray	647
6.115.3.13looseUnmarshalLong	647
6.115.3.14looseUnmarshalNestedObject	648
6.115.3.15looseUnmarshalString	648

6.115.3.16	readAsciiString	648
6.115.3.17	rightMarshal1	649
6.115.3.18	rightMarshal2	649
6.115.3.19	rightMarshalBrokerError1	649
6.115.3.20	rightMarshalBrokerError2	650
6.115.3.21	rightMarshalCachedObject1	650
6.115.3.22	rightMarshalCachedObject2	651
6.115.3.23	rightMarshalLong1	651
6.115.3.24	rightMarshalLong2	651
6.115.3.25	rightMarshalNestedObject1	652
6.115.3.26	rightMarshalNestedObject2	652
6.115.3.27	rightMarshalObjectArray1	653
6.115.3.28	rightMarshalObjectArray2	653
6.115.3.29	rightMarshalString1	654
6.115.3.30	rightMarshalString2	654
6.115.3.31	rightUnmarshal	654
6.115.3.32	rightUnmarshalBrokerError	655
6.115.3.33	rightUnmarshalByteArray	655
6.115.3.34	rightUnmarshalCachedObject	656
6.115.3.35	rightUnmarshalConstByteArray	656
6.115.3.36	rightUnmarshalLong	656
6.115.3.37	rightUnmarshalNestedObject	657
6.115.3.38	rightUnmarshalString	657
6.115.3.39	toHexFromBytes	658
6.115.3.40	toString	658
6.115.3.41	toString	658
6.115.3.42	toString	658
6.116	activemq::commands::BaseDataStructure Class Reference	659
6.116.1	Constructor & Destructor Documentation	660
6.116.1.1	~BaseDataStructure	660
6.116.2	Member Function Documentation	660
6.116.2.1	afterMarshal	660
6.116.2.2	afterUnmarshal	660
6.116.2.3	beforeMarshal	660
6.116.2.4	beforeUnmarshal	661
6.116.2.5	copyDataStructure	661

6.116.2.6 equals	661
6.116.2.7 getMarshaledForm	661
6.116.2.8 isMarshalAware	662
6.116.2.9 setMarshaledForm	662
6.116.2.10oString	662
6.117binary_function Class Reference	663
6.118decaf::net::BindException Class Reference	663
6.118.1 Constructor & Destructor Documentation	664
6.118.1.1 BindException	664
6.118.1.2 BindException	664
6.118.1.3 BindException	664
6.118.1.4 BindException	665
6.118.1.5 BindException	665
6.118.1.6 BindException	665
6.118.1.7 ~BindException	665
6.118.2 Member Function Documentation	665
6.118.2.1 clone	665
6.119decaf::io::BlockingByteArrayInputStream Class Reference	666
6.119.1 Detailed Description	667
6.119.2 Constructor & Destructor Documentation	668
6.119.2.1 BlockingByteArrayInputStream	668
6.119.2.2 BlockingByteArrayInputStream	668
6.119.2.3 ~BlockingByteArrayInputStream	668
6.119.3 Member Function Documentation	668
6.119.3.1 available	668
6.119.3.2 close	668
6.119.3.3 lock	669
6.119.3.4 mark	669
6.119.3.5 markSupported	669
6.119.3.6 notify	669
6.119.3.7 notifyAll	670
6.119.3.8 read	670
6.119.3.9 read	670
6.119.3.10reset	671
6.119.3.11setByteArray	671
6.119.3.12skip	671

6.119.3.13	tryLock	672
6.119.3.14	unlock	672
6.119.3.15	wait	672
6.119.3.16	wait	673
6.119.3.17	wait	673
6.120	decaf::lang::Boolean Class Reference	674
6.120.1	Constructor & Destructor Documentation	675
6.120.1.1	Boolean	675
6.120.1.2	Boolean	675
6.120.1.3	~Boolean	675
6.120.2	Member Function Documentation	675
6.120.2.1	booleanValue	675
6.120.2.2	compareTo	675
6.120.2.3	compareTo	676
6.120.2.4	equals	676
6.120.2.5	equals	676
6.120.2.6	operator<	676
6.120.2.7	operator<	676
6.120.2.8	operator==	677
6.120.2.9	operator==	677
6.120.2.10	parseBoolean	677
6.120.2.11	toString	678
6.120.2.12	toString	678
6.120.2.13	valueOf	678
6.120.2.14	valueOf	678
6.120.3	Field Documentation	678
6.120.3.1	_FALSE	678
6.120.3.2	_TRUE	678
6.121	activemq::commands::BooleanExpression Class Reference	679
6.121.1	Constructor & Destructor Documentation	679
6.121.1.1	BooleanExpression	679
6.121.1.2	~BooleanExpression	679
6.121.2	Member Function Documentation	679
6.121.2.1	cloneDataStructure	679
6.121.2.2	copyDataStructure	680
6.121.2.3	equals	680

6.121.2.4 toString	680
6.122activemq::wireformat::openwire::utils::BooleanStream Class Reference	680
6.122.1 Detailed Description	681
6.122.2 Constructor & Destructor Documentation	682
6.122.2.1 BooleanStream	682
6.122.2.2 ~BooleanStream	682
6.122.3 Member Function Documentation	682
6.122.3.1 clear	682
6.122.3.2 marshal	682
6.122.3.3 marshal	682
6.122.3.4 marshalledSize	682
6.122.3.5 readBoolean	682
6.122.3.6 unmarshal	683
6.122.3.7 writeBoolean	683
6.123decaf::util::concurrent::BrokenBarrierException Class Reference	683
6.123.1 Constructor & Destructor Documentation	684
6.123.1.1 BrokenBarrierException	684
6.123.1.2 BrokenBarrierException	684
6.123.1.3 BrokenBarrierException	684
6.123.1.4 BrokenBarrierException	684
6.123.1.5 BrokenBarrierException	685
6.123.1.6 BrokenBarrierException	685
6.123.1.7 ~BrokenBarrierException	685
6.123.2 Member Function Documentation	685
6.123.2.1 clone	685
6.124activemq::commands::BrokerError Class Reference	686
6.124.1 Detailed Description	687
6.124.2 Constructor & Destructor Documentation	687
6.124.2.1 BrokerError	687
6.124.2.2 ~BrokerError	687
6.124.3 Member Function Documentation	687
6.124.3.1 cloneDataStructure	687
6.124.3.2 copyDataStructure	687
6.124.3.3 getCause	688
6.124.3.4 getDataStructureType	688
6.124.3.5 getExceptionClass	688

6.124.3.6	getMessage	688
6.124.3.7	getStackTraceElements	688
6.124.3.8	setCause	689
6.124.3.9	setExceptionClass	689
6.124.3.10	setMessage	689
6.124.3.11	setStackTraceElements	689
6.124.3.12	visit	689
6.125	activemq::exceptions::BrokerException Class Reference	690
6.125.1	Constructor & Destructor Documentation	690
6.125.1.1	BrokerException	690
6.125.1.2	BrokerException	690
6.125.1.3	BrokerException	690
6.125.1.4	BrokerException	690
6.125.1.5	BrokerException	690
6.125.1.6	~BrokerException	691
6.125.2	Member Function Documentation	691
6.125.2.1	clone	691
6.126	activemq::commands::BrokerId Class Reference	691
6.126.1	Member Typedef Documentation	692
6.126.1.1	COMPARATOR	692
6.126.2	Constructor & Destructor Documentation	692
6.126.2.1	BrokerId	692
6.126.2.2	BrokerId	692
6.126.2.3	~BrokerId	692
6.126.3	Member Function Documentation	692
6.126.3.1	cloneDataStructure	692
6.126.3.2	compareTo	692
6.126.3.3	copyDataStructure	692
6.126.3.4	equals	693
6.126.3.5	equals	693
6.126.3.6	getDataStructureType	693
6.126.3.7	getValue	693
6.126.3.8	getValue	693
6.126.3.9	operator<	693
6.126.3.10	operator=	693
6.126.3.11	operator==	693

6.126.3.12set Value	693
6.126.3.13oString	693
6.126.4 Field Documentation	694
6.126.4.1 ID_BROKERID	694
6.126.4.2 value	694
6.127activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller Class Reference	694
6.127.1 Detailed Description	695
6.127.2 Constructor & Destructor Documentation	695
6.127.2.1 BrokerIdMarshaller	695
6.127.2.2 ~BrokerIdMarshaller	695
6.127.3 Member Function Documentation	695
6.127.3.1 createObject	695
6.127.3.2 getDataStructureType	695
6.127.3.3 looseMarshal	696
6.127.3.4 looseUnmarshal	696
6.127.3.5 tightMarshal1	696
6.127.3.6 tightMarshal2	697
6.127.3.7 tightUnmarshal	697
6.128activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller Class Reference	698
6.128.1 Detailed Description	699
6.128.2 Constructor & Destructor Documentation	699
6.128.2.1 BrokerIdMarshaller	699
6.128.2.2 ~BrokerIdMarshaller	699
6.128.3 Member Function Documentation	699
6.128.3.1 createObject	699
6.128.3.2 getDataStructureType	699
6.128.3.3 looseMarshal	699
6.128.3.4 looseUnmarshal	700
6.128.3.5 tightMarshal1	700
6.128.3.6 tightMarshal2	701
6.128.3.7 tightUnmarshal	701
6.129activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller Class Reference	701
6.129.1 Detailed Description	702
6.129.2 Constructor & Destructor Documentation	703
6.129.2.1 BrokerIdMarshaller	703
6.129.2.2 ~BrokerIdMarshaller	703

6.129.3 Member Function Documentation	703
6.129.3.1 createObject	703
6.129.3.2 getDataStructureType	703
6.129.3.3 looseMarshal	703
6.129.3.4 looseUnmarshal	704
6.129.3.5 tightMarshal1	704
6.129.3.6 tightMarshal2	704
6.129.3.7 tightUnmarshal	705
6.130activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller Class Reference .	705
6.130.1 Detailed Description	706
6.130.2 Constructor & Destructor Documentation	707
6.130.2.1 BrokerIdMarshaller	707
6.130.2.2 ~BrokerIdMarshaller	707
6.130.3 Member Function Documentation	707
6.130.3.1 createObject	707
6.130.3.2 getDataStructureType	707
6.130.3.3 looseMarshal	707
6.130.3.4 looseUnmarshal	708
6.130.3.5 tightMarshal1	708
6.130.3.6 tightMarshal2	708
6.130.3.7 tightUnmarshal	709
6.131activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller Class Reference .	709
6.131.1 Detailed Description	710
6.131.2 Constructor & Destructor Documentation	711
6.131.2.1 BrokerIdMarshaller	711
6.131.2.2 ~BrokerIdMarshaller	711
6.131.3 Member Function Documentation	711
6.131.3.1 createObject	711
6.131.3.2 getDataStructureType	711
6.131.3.3 looseMarshal	711
6.131.3.4 looseUnmarshal	712
6.131.3.5 tightMarshal1	712
6.131.3.6 tightMarshal2	712
6.131.3.7 tightUnmarshal	713
6.132activemq::commands::BrokerInfo Class Reference	713
6.132.1 Constructor & Destructor Documentation	715

6.132.1.1 BrokerInfo	715
6.132.1.2 BrokerInfo	715
6.132.1.3 ~BrokerInfo	715
6.132.2 Member Function Documentation	715
6.132.2.1 cloneDataStructure	715
6.132.2.2 copyDataStructure	716
6.132.2.3 equals	716
6.132.2.4 getBrokerId	717
6.132.2.5 getBrokerId	717
6.132.2.6 getBrokerName	717
6.132.2.7 getBrokerName	717
6.132.2.8 getBrokerUploadUrl	717
6.132.2.9 getBrokerUploadUrl	717
6.132.2.10getBrokerURL	717
6.132.2.11getBrokerURL	717
6.132.2.12getConnectionId	717
6.132.2.13getDataStructureType	717
6.132.2.14getNetworkProperties	718
6.132.2.15getNetworkProperties	718
6.132.2.16getPeerBrokerInfos	718
6.132.2.17getPeerBrokerInfos	718
6.132.2.18sBrokerInfo	718
6.132.2.19sDuplexConnection	719
6.132.2.20sFault Tolerant Configuration	719
6.132.2.21sMasterBroker	719
6.132.2.22sNetworkConnection	719
6.132.2.23sSlaveBroker	719
6.132.2.24operator=	719
6.132.2.25setBrokerId	719
6.132.2.26setBrokerName	719
6.132.2.27setBrokerUploadUrl	719
6.132.2.28setBrokerURL	719
6.132.2.29setConnectionId	719
6.132.2.30setDuplexConnection	719
6.132.2.31setFault Tolerant Configuration	719
6.132.2.32setMasterBroker	719

6.132.2.33	setNetworkConnection	719
6.132.2.34	setNetworkProperties	719
6.132.2.35	setPeerBrokerInfos	719
6.132.2.36	setSlaveBroker	719
6.132.2.37	toString	719
6.132.2.38	visit	720
6.132.3	Field Documentation	721
6.132.3.1	brokerId	721
6.132.3.2	brokerName	721
6.132.3.3	brokerUploadUrl	721
6.132.3.4	brokerURL	721
6.132.3.5	connectionId	721
6.132.3.6	duplexConnection	721
6.132.3.7	faultTolerantConfiguration	721
6.132.3.8	ID_BROKERINFO	721
6.132.3.9	masterBroker	721
6.132.3.10	networkConnection	721
6.132.3.11	networkProperties	721
6.132.3.12	peerBrokerInfos	721
6.132.3.13	slaveBroker	721
6.133	activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller Class Reference	721
6.133.1	Detailed Description	722
6.133.2	Constructor & Destructor Documentation	723
6.133.2.1	BrokerInfoMarshaller	723
6.133.2.2	~BrokerInfoMarshaller	723
6.133.3	Member Function Documentation	723
6.133.3.1	createObject	723
6.133.3.2	getDataStructureType	723
6.133.3.3	looseMarshal	723
6.133.3.4	looseUnmarshal	724
6.133.3.5	tightMarshal1	724
6.133.3.6	tightMarshal2	724
6.133.3.7	tightUnmarshal	725
6.134	activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller Class Reference	725
6.134.1	Detailed Description	726
6.134.2	Constructor & Destructor Documentation	727

6.134.2.1 BrokerInfoMarshaller	727
6.134.2.2 ~BrokerInfoMarshaller	727
6.134.3 Member Function Documentation	727
6.134.3.1 createObject	727
6.134.3.2 getDataStructureType	727
6.134.3.3 looseMarshal	727
6.134.3.4 looseUnmarshal	728
6.134.3.5 tightMarshal1	728
6.134.3.6 tightMarshal2	728
6.134.3.7 tightUnmarshal	729
6.135activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller Class Reference	729
6.135.1 Detailed Description	730
6.135.2 Constructor & Destructor Documentation	731
6.135.2.1 BrokerInfoMarshaller	731
6.135.2.2 ~BrokerInfoMarshaller	731
6.135.3 Member Function Documentation	731
6.135.3.1 createObject	731
6.135.3.2 getDataStructureType	731
6.135.3.3 looseMarshal	731
6.135.3.4 looseUnmarshal	732
6.135.3.5 tightMarshal1	732
6.135.3.6 tightMarshal2	732
6.135.3.7 tightUnmarshal	733
6.136activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller Class Reference	733
6.136.1 Detailed Description	734
6.136.2 Constructor & Destructor Documentation	735
6.136.2.1 BrokerInfoMarshaller	735
6.136.2.2 ~BrokerInfoMarshaller	735
6.136.3 Member Function Documentation	735
6.136.3.1 createObject	735
6.136.3.2 getDataStructureType	735
6.136.3.3 looseMarshal	735
6.136.3.4 looseUnmarshal	736
6.136.3.5 tightMarshal1	736
6.136.3.6 tightMarshal2	736
6.136.3.7 tightUnmarshal	737

6.137	activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller Class Reference	737
6.137.1	Detailed Description	738
6.137.2	Constructor & Destructor Documentation	739
6.137.2.1	BrokerInfoMarshaller	739
6.137.2.2	~BrokerInfoMarshaller	739
6.137.3	Member Function Documentation	739
6.137.3.1	createObject	739
6.137.3.2	getDataStructureType	739
6.137.3.3	looseMarshal	739
6.137.3.4	looseUnmarshal	740
6.137.3.5	tightMarshal1	740
6.137.3.6	tightMarshal2	740
6.137.3.7	tightUnmarshal	741
6.138	decaf::nio::Buffer Class Reference	741
6.138.1	Detailed Description	743
6.138.2	Constructor & Destructor Documentation	744
6.138.2.1	Buffer	744
6.138.2.2	Buffer	744
6.138.2.3	~Buffer	744
6.138.3	Member Function Documentation	744
6.138.3.1	capacity	744
6.138.3.2	clear	744
6.138.3.3	flip	744
6.138.3.4	hasRemaining	745
6.138.3.5	isReadOnly	745
6.138.3.6	limit	745
6.138.3.7	limit	745
6.138.3.8	mark	746
6.138.3.9	position	746
6.138.3.10	position	746
6.138.3.11	remaining	746
6.138.3.12	reset	747
6.138.3.13	rewind	747
6.138.4	Field Documentation	747
6.138.4.1	_capacity	747
6.138.4.2	_limit	747

6.138.4.3	_mark	747
6.138.4.4	_markSet	747
6.138.4.5	_position	747
6.139	decaf::io::BufferedInputStream Class Reference	747
6.139.1	Detailed Description	748
6.139.2	Constructor & Destructor Documentation	748
6.139.2.1	BufferedInputStream	748
6.139.2.2	BufferedInputStream	749
6.139.2.3	~BufferedInputStream	749
6.139.3	Member Function Documentation	749
6.139.3.1	available	749
6.139.3.2	close	749
6.139.3.3	mark	750
6.139.3.4	markSupported	750
6.139.3.5	read	750
6.139.3.6	read	751
6.139.3.7	reset	751
6.139.3.8	skip	751
6.140	decaf::io::BufferedOutputStream Class Reference	752
6.140.1	Detailed Description	753
6.140.2	Constructor & Destructor Documentation	753
6.140.2.1	BufferedOutputStream	753
6.140.2.2	BufferedOutputStream	753
6.140.2.3	~BufferedOutputStream	753
6.140.3	Member Function Documentation	753
6.140.3.1	close	753
6.140.3.2	flush	753
6.140.3.3	write	754
6.140.3.4	write	754
6.140.3.5	write	754
6.141	decaf::net::BufferedSocket Class Reference	755
6.141.1	Detailed Description	756
6.141.2	Constructor & Destructor Documentation	756
6.141.2.1	BufferedSocket	756
6.141.2.2	~BufferedSocket	757
6.141.3	Member Function Documentation	757

6.141.3.1 close	757
6.141.3.2 connect	757
6.141.3.3 getInputStream	757
6.141.3.4 getKeepAlive	757
6.141.3.5 getOutputStream	758
6.141.3.6 getReceiveBufferSize	758
6.141.3.7 getReuseAddress	758
6.141.3.8 getSendBufferSize	759
6.141.3.9 getSoLinger	759
6.141.3.10getSoTimeout	759
6.141.3.11isConnected	759
6.141.3.12setKeepAlive	760
6.141.3.13setReceiveBufferSize	760
6.141.3.14setReuseAddress	760
6.141.3.15setSendBufferSize	761
6.141.3.16setSoLinger	761
6.141.3.17setSoTimeout	761
6.142decaf::internal::nio::BufferFactory Class Reference	762
6.142.1 Detailed Description	763
6.142.2 Constructor & Destructor Documentation	764
6.142.2.1 ~BufferFactory	764
6.142.3 Member Function Documentation	764
6.142.3.1 createByteBuffer	764
6.142.3.2 createByteBuffer	764
6.142.3.3 createByteBuffer	764
6.142.3.4 createCharBuffer	765
6.142.3.5 createCharBuffer	765
6.142.3.6 createCharBuffer	765
6.142.3.7 createDoubleBuffer	766
6.142.3.8 createDoubleBuffer	766
6.142.3.9 createDoubleBuffer	767
6.142.3.10createFloatBuffer	767
6.142.3.11createFloatBuffer	767
6.142.3.12reateFloatBuffer	768
6.142.3.13reateIntBuffer	768
6.142.3.14reateIntBuffer	768

6.142.3.15	createIntBuffer	769
6.142.3.16	createLongBuffer	769
6.142.3.17	createLongBuffer	769
6.142.3.18	createLongBuffer	770
6.142.3.19	createShortBuffer	770
6.142.3.20	createShortBuffer	770
6.142.3.21	createShortBuffer	771
6.143	decaf::nio::BufferOverflowException Class Reference	771
6.143.1	Constructor & Destructor Documentation	772
6.143.1.1	BufferOverflowException	772
6.143.1.2	BufferOverflowException	772
6.143.1.3	BufferOverflowException	772
6.143.1.4	BufferOverflowException	772
6.143.1.5	BufferOverflowException	773
6.143.1.6	BufferOverflowException	773
6.143.1.7	~BufferOverflowException	773
6.143.2	Member Function Documentation	773
6.143.2.1	clone	773
6.144	decaf::nio::BufferUnderflowException Class Reference	774
6.144.1	Constructor & Destructor Documentation	774
6.144.1.1	BufferUnderflowException	774
6.144.1.2	BufferUnderflowException	774
6.144.1.3	BufferUnderflowException	775
6.144.1.4	BufferUnderflowException	775
6.144.1.5	BufferUnderflowException	775
6.144.1.6	BufferUnderflowException	775
6.144.1.7	~BufferUnderflowException	776
6.144.2	Member Function Documentation	776
6.144.2.1	clone	776
6.145	decaf::lang::Byte Class Reference	776
6.145.1	Constructor & Destructor Documentation	778
6.145.1.1	Byte	778
6.145.1.2	Byte	778
6.145.1.3	~Byte	778
6.145.2	Member Function Documentation	778
6.145.2.1	byteValue	778

6.145.2.2 compareTo	779
6.145.2.3 compareTo	779
6.145.2.4 decode	779
6.145.2.5 doubleValue	780
6.145.2.6 equals	780
6.145.2.7 equals	780
6.145.2.8 floatValue	780
6.145.2.9 intValue	780
6.145.2.10 longValue	780
6.145.2.11 operator<	781
6.145.2.12 operator<	781
6.145.2.13 operator==	781
6.145.2.14 operator==	781
6.145.2.15 parseByte	782
6.145.2.16 parseByte	782
6.145.2.17 shortValue	783
6.145.2.18 toString	783
6.145.2.19 toString	783
6.145.2.20 valueOf	783
6.145.2.21 valueOf	783
6.145.2.22 valueOf	784
6.145.3 Field Documentation	784
6.145.3.1 MAX_VALUE	784
6.145.3.2 MIN_VALUE	784
6.145.3.3 SIZE	784
6.146 decaf::internal::util::ByteArrayAdapter Class Reference	784
6.146.1 Detailed Description	789
6.146.2 Constructor & Destructor Documentation	789
6.146.2.1 ByteArrayAdapter	789
6.146.2.2 ByteArrayAdapter	789
6.146.2.3 ByteArrayAdapter	789
6.146.2.4 ByteArrayAdapter	790
6.146.2.5 ByteArrayAdapter	790
6.146.2.6 ByteArrayAdapter	790
6.146.2.7 ByteArrayAdapter	791
6.146.2.8 ByteArrayAdapter	791

6.146.2.9 ~ByteArrayAdapter	791
6.146.3 Member Function Documentation	791
6.146.3.1 clear	791
6.146.3.2 get	792
6.146.3.3 getByteArray	792
6.146.3.4 getCapacity	792
6.146.3.5 getChar	792
6.146.3.6 getCharArray	793
6.146.3.7 getCharCapacity	793
6.146.3.8 getDouble	793
6.146.3.9 getDoubleArray	794
6.146.3.10 getDoubleAt	794
6.146.3.11 getDoubleCapacity	794
6.146.3.12 getFloat	794
6.146.3.13 getFloatArray	795
6.146.3.14 getFloatAt	795
6.146.3.15 getFloatCapacity	795
6.146.3.16 getInt	796
6.146.3.17 getIntArray	796
6.146.3.18 getIntAt	796
6.146.3.19 getIntCapacity	797
6.146.3.20 getLong	797
6.146.3.21 getLongArray	797
6.146.3.22 getLongAt	797
6.146.3.23 getLongCapacity	798
6.146.3.24 getShort	798
6.146.3.25 getShortArray	798
6.146.3.26 getShortAt	799
6.146.3.27 getShortCapacity	799
6.146.3.28 operator[]	799
6.146.3.29 operator[]	800
6.146.3.30 put	800
6.146.3.31 putChar	800
6.146.3.32 putDouble	801
6.146.3.33 putDoubleAt	801
6.146.3.34 putFloat	801

6.146.3.35	putFloatAt	802
6.146.3.36	putInt	802
6.146.3.37	putIntAt	803
6.146.3.38	putLong	803
6.146.3.39	putLongAt	803
6.146.3.40	putShort	804
6.146.3.41	putShortAt	804
6.146.3.42	read	805
6.146.3.43	resize	805
6.146.3.44	write	805
6.147	decaf::internal::nio::ByteBuffer Class Reference	806
6.147.1	Detailed Description	810
6.147.2	Constructor & Destructor Documentation	811
6.147.2.1	ByteBuffer	811
6.147.2.2	ByteBuffer	811
6.147.2.3	ByteBuffer	811
6.147.2.4	ByteBuffer	812
6.147.2.5	~ByteBuffer	812
6.147.3	Member Function Documentation	812
6.147.3.1	array	812
6.147.3.2	arrayOffset	812
6.147.3.3	asCharBuffer	813
6.147.3.4	asDoubleBuffer	813
6.147.3.5	asFloatBuffer	813
6.147.3.6	asIntBuffer	814
6.147.3.7	asLongBuffer	814
6.147.3.8	asReadOnlyBuffer	814
6.147.3.9	asShortBuffer	815
6.147.3.10	compact	815
6.147.3.11	duplicate	816
6.147.3.12	get	816
6.147.3.13	get	816
6.147.3.14	getChar	817
6.147.3.15	getChar	817
6.147.3.16	getDouble	817
6.147.3.17	getDouble	818

6.147.3.18	getFloat	818
6.147.3.19	getFloat	818
6.147.3.20	getInt	819
6.147.3.21	getInt	819
6.147.3.22	getLong	819
6.147.3.23	getLong	820
6.147.3.24	getShort	820
6.147.3.25	getShort	820
6.147.3.26	hasArray	821
6.147.3.27	isReadOnly	821
6.147.3.28	put	821
6.147.3.29	put	821
6.147.3.30	putChar	822
6.147.3.31	putChar	822
6.147.3.32	putDouble	823
6.147.3.33	putDouble	823
6.147.3.34	putFloat	824
6.147.3.35	putFloat	824
6.147.3.36	putInt	824
6.147.3.37	putInt	825
6.147.3.38	putLong	825
6.147.3.39	putLong	826
6.147.3.40	putShort	826
6.147.3.41	putShort	827
6.147.3.42	setReadOnly	827
6.147.3.43	slice	827
6.148	decaf::io::ByteArrayInputStream Class Reference	828
6.148.1	Detailed Description	829
6.148.2	Constructor & Destructor Documentation	830
6.148.2.1	ByteArrayInputStream	830
6.148.2.2	ByteArrayInputStream	830
6.148.2.3	ByteArrayInputStream	830
6.148.2.4	~ByteArrayInputStream	830
6.148.3	Member Function Documentation	830
6.148.3.1	available	830
6.148.3.2	close	830

6.148.3.3 lock	831
6.148.3.4 mark	831
6.148.3.5 markSupported	831
6.148.3.6 notify	831
6.148.3.7 notifyAll	832
6.148.3.8 read	832
6.148.3.9 read	832
6.148.3.10 reset	833
6.148.3.11 setBuffer	833
6.148.3.12 setByteArray	833
6.148.3.13 skip	833
6.148.3.14 tryLock	834
6.148.3.15 unlock	834
6.148.3.16 wait	834
6.148.3.17 wait	835
6.148.3.18 wait	835
6.149 decaf::io::ByteArrayOutputStream Class Reference	836
6.149.1 Constructor & Destructor Documentation	837
6.149.1.1 ByteArrayOutputStream	837
6.149.1.2 ByteArrayOutputStream	838
6.149.1.3 ~ByteArrayOutputStream	838
6.149.2 Member Function Documentation	838
6.149.2.1 close	838
6.149.2.2 flush	838
6.149.2.3 lock	838
6.149.2.4 notify	838
6.149.2.5 notifyAll	839
6.149.2.6 reset	839
6.149.2.7 setBuffer	839
6.149.2.8 size	840
6.149.2.9 toByteArray	840
6.149.2.10 toByteArrayRef	840
6.149.2.11 toString	840
6.149.2.12 tryLock	840
6.149.2.13 unlock	841
6.149.2.14 wait	841

6.149.2.15wait	841
6.149.2.16wait	842
6.149.2.17write	842
6.149.2.18write	842
6.149.2.19write	843
6.149.2.20writeTo	843
6.150decaf::internal::nio::ByteArrayPerspective Class Reference	843
6.150.1 Detailed Description	844
6.150.2 Constructor & Destructor Documentation	845
6.150.2.1 ByteArrayPerspective	845
6.150.2.2 ByteArrayPerspective	845
6.150.2.3 ByteArrayPerspective	845
6.150.2.4 ByteArrayPerspective	846
6.150.2.5 ByteArrayPerspective	846
6.150.2.6 ByteArrayPerspective	846
6.150.2.7 ByteArrayPerspective	847
6.150.2.8 ByteArrayPerspective	847
6.150.2.9 ~ByteArrayPerspective	847
6.150.3 Member Function Documentation	847
6.150.3.1 getReferences	847
6.150.3.2 returnRef	847
6.150.3.3 takeRef	848
6.151decaf::nio::ByteBuffer Class Reference	848
6.151.1 Detailed Description	852
6.151.2 Constructor & Destructor Documentation	853
6.151.2.1 ByteBuffer	853
6.151.2.2 ~ByteBuffer	854
6.151.3 Member Function Documentation	854
6.151.3.1 allocate	854
6.151.3.2 array	854
6.151.3.3 arrayOffset	854
6.151.3.4 asCharBuffer	855
6.151.3.5 asDoubleBuffer	855
6.151.3.6 asFloatBuffer	855
6.151.3.7 asIntBuffer	856
6.151.3.8 asLongBuffer	856

6.151.3.9 asReadOnlyBuffer	856
6.151.3.10 asShortBuffer	857
6.151.3.11 compact	857
6.151.3.12 compareTo	857
6.151.3.13 duplicate	858
6.151.3.14 equals	858
6.151.3.15 get	858
6.151.3.16 get	859
6.151.3.17 get	859
6.151.3.18 get	859
6.151.3.19 getChar	860
6.151.3.20 getChar	860
6.151.3.21 getDouble	861
6.151.3.22 getDouble	861
6.151.3.23 getFloat	861
6.151.3.24 getFloat	862
6.151.3.25 getInt	862
6.151.3.26 getInt	862
6.151.3.27 getLong	863
6.151.3.28 getLong	863
6.151.3.29 getShort	863
6.151.3.30 getShort	864
6.151.3.31 hasArray	864
6.151.3.32 isReadOnly	864
6.151.3.33 operator<	864
6.151.3.34 operator==	865
6.151.3.35 put	865
6.151.3.36 put	865
6.151.3.37 put	866
6.151.3.38 put	866
6.151.3.39 put	867
6.151.3.40 putChar	867
6.151.3.41 putChar	868
6.151.3.42 putDouble	868
6.151.3.43 putDouble	869
6.151.3.44 putFloat	869

6.151.3.45	putFloat	869
6.151.3.46	putInt	870
6.151.3.47	putInt	870
6.151.3.48	putLong	871
6.151.3.49	putLong	871
6.151.3.50	putShort	872
6.151.3.51	putShort	872
6.151.3.52	slice	872
6.151.3.53	toString	873
6.151.3.54	wrap	873
6.151.3.55	wrap	873
6.152	cms::BytesMessage Class Reference	874
6.152.1	Detailed Description	876
6.152.2	Constructor & Destructor Documentation	877
6.152.2.1	~BytesMessage	877
6.152.3	Member Function Documentation	877
6.152.3.1	clone	877
6.152.3.2	getBodyBytes	877
6.152.3.3	getBodyLength	878
6.152.3.4	readBoolean	878
6.152.3.5	readByte	878
6.152.3.6	readBytes	879
6.152.3.7	readBytes	879
6.152.3.8	readChar	880
6.152.3.9	readDouble	880
6.152.3.10	readFloat	881
6.152.3.11	readInt	881
6.152.3.12	readLong	881
6.152.3.13	readShort	882
6.152.3.14	readString	882
6.152.3.15	readUnsignedShort	882
6.152.3.16	readUTF	883
6.152.3.17	reset	883
6.152.3.18	setBodyBytes	883
6.152.3.19	writeBoolean	883
6.152.3.20	writeByte	884

6.152.3.21	writeBytes	884
6.152.3.22	writeBytes	884
6.152.3.23	writeChar	885
6.152.3.24	writeDouble	885
6.152.3.25	writeFloat	885
6.152.3.26	writeInt	886
6.152.3.27	writeLong	886
6.152.3.28	writeShort	886
6.152.3.29	writeString	887
6.152.3.30	writeUnsignedShort	887
6.152.3.31	writeUTF	887
6.153	activemq::cmsutil::CachedConsumer Class Reference	888
6.153.1	Detailed Description	888
6.153.2	Constructor & Destructor Documentation	889
6.153.2.1	CachedConsumer	889
6.153.2.2	~CachedConsumer	889
6.153.3	Member Function Documentation	889
6.153.3.1	close	889
6.153.3.2	getMessageListener	889
6.153.3.3	getMessageSelector	889
6.153.3.4	receive	889
6.153.3.5	receive	890
6.153.3.6	receiveNoWait	890
6.153.3.7	setMessageListener	890
6.154	activemq::cmsutil::CachedProducer Class Reference	891
6.154.1	Detailed Description	892
6.154.2	Constructor & Destructor Documentation	892
6.154.2.1	CachedProducer	892
6.154.2.2	~CachedProducer	892
6.154.3	Member Function Documentation	892
6.154.3.1	close	892
6.154.3.2	getDeliveryMode	892
6.154.3.3	getDisableMessageID	893
6.154.3.4	getDisableMessageTimeStamp	893
6.154.3.5	getPriority	893
6.154.3.6	getTimeToLive	894

6.154.3.7 send	894
6.154.3.8 send	894
6.154.3.9 send	895
6.154.3.10 send	895
6.154.3.11 setDeliveryMode	896
6.154.3.12 setDisableMessageID	896
6.154.3.13 setDisableMessageTimeStamp	896
6.154.3.14 setPriority	897
6.154.3.15 setTimeToLive	897
6.155 decaf::util::concurrent::Callable< V > Class Template Reference	897
6.155.1 Detailed Description	898
6.155.2 Constructor & Destructor Documentation	898
6.155.2.1 ~Callable	898
6.155.3 Member Function Documentation	898
6.155.3.1 call	898
6.156 decaf::util::concurrent::CancellationException Class Reference	898
6.156.1 Constructor & Destructor Documentation	899
6.156.1.1 CancellationException	899
6.156.1.2 CancellationException	899
6.156.1.3 CancellationException	899
6.156.1.4 CancellationException	900
6.156.1.5 CancellationException	900
6.156.1.6 CancellationException	900
6.156.1.7 ~CancellationException	900
6.156.2 Member Function Documentation	900
6.156.2.1 clone	900
6.157 decaf::security::cert::Certificate Class Reference	901
6.157.1 Detailed Description	902
6.157.2 Constructor & Destructor Documentation	902
6.157.2.1 ~Certificate	902
6.157.3 Member Function Documentation	902
6.157.3.1 equals	902
6.157.3.2 getEncoded	902
6.157.3.3 getPublicKey	902
6.157.3.4 getPublicKey	903
6.157.3.5 getType	903

6.157.3.6 toString	903
6.157.3.7 verify	903
6.157.3.8 verify	904
6.158decaf::security::cert::CertificateEncodingException Class Reference	904
6.158.1 Constructor & Destructor Documentation	905
6.158.1.1 CertificateEncodingException	905
6.158.1.2 CertificateEncodingException	905
6.158.1.3 CertificateEncodingException	905
6.158.1.4 CertificateEncodingException	906
6.158.1.5 ~CertificateEncodingException	906
6.158.2 Member Function Documentation	906
6.158.2.1 clone	906
6.159decaf::security::cert::CertificateException Class Reference	906
6.159.1 Constructor & Destructor Documentation	907
6.159.1.1 CertificateException	907
6.159.1.2 CertificateException	907
6.159.1.3 CertificateException	907
6.159.1.4 CertificateException	908
6.159.1.5 ~CertificateException	908
6.159.2 Member Function Documentation	908
6.159.2.1 clone	908
6.160decaf::security::cert::CertificateExpiredException Class Reference	908
6.160.1 Constructor & Destructor Documentation	909
6.160.1.1 CertificateExpiredException	909
6.160.1.2 CertificateExpiredException	909
6.160.1.3 CertificateExpiredException	909
6.160.1.4 CertificateExpiredException	910
6.160.1.5 ~CertificateExpiredException	910
6.160.2 Member Function Documentation	910
6.160.2.1 clone	910
6.161decaf::security::cert::CertificateNotYetValidException Class Reference	910
6.161.1 Constructor & Destructor Documentation	911
6.161.1.1 CertificateNotYetValidException	911
6.161.1.2 CertificateNotYetValidException	911
6.161.1.3 CertificateNotYetValidException	911
6.161.1.4 CertificateNotYetValidException	912

6.161.1.5 ~CertificateNotYetValidException	912
6.161.2 Member Function Documentation	912
6.161.2.1 clone	912
6.162decaf::security::cert::CertificateParsingException Class Reference	912
6.162.1 Constructor & Destructor Documentation	913
6.162.1.1 CertificateParsingException	913
6.162.1.2 CertificateParsingException	913
6.162.1.3 CertificateParsingException	913
6.162.1.4 CertificateParsingException	914
6.162.1.5 ~CertificateParsingException	914
6.162.2 Member Function Documentation	914
6.162.2.1 clone	914
6.163decaf::lang::Character Class Reference	914
6.163.1 Constructor & Destructor Documentation	916
6.163.1.1 Character	916
6.163.2 Member Function Documentation	917
6.163.2.1 byteValue	917
6.163.2.2 compareTo	917
6.163.2.3 compareTo	917
6.163.2.4 digit	917
6.163.2.5 doubleValue	918
6.163.2.6 equals	918
6.163.2.7 equals	918
6.163.2.8 float Value	918
6.163.2.9 int Value	918
6.163.2.10sDigit	919
6.163.2.11sISOControl	919
6.163.2.12sLetter	919
6.163.2.13sLetterOrDigit	919
6.163.2.14sLowerCase	919
6.163.2.15sUpperCase	919
6.163.2.16sWhitespace	919
6.163.2.17ongValue	919
6.163.2.18operator<	920
6.163.2.19operator<	920
6.163.2.20operator==	920

6.163.2.21operator==	920
6.163.2.22shortValue	921
6.163.2.23toString	921
6.163.2.24valueOf	921
6.163.3Field Documentation	921
6.163.3.1 MAX_RADIX	921
6.163.3.2 MAX_VALUE	921
6.163.3.3 MIN_RADIX	921
6.163.3.4 MIN_VALUE	921
6.163.3.5 SIZE	922
6.164decaf::internal::nio::CharArrayBuffer Class Reference	922
6.164.1 Constructor & Destructor Documentation	924
6.164.1.1 CharArrayBuffer	924
6.164.1.2 CharArrayBuffer	924
6.164.1.3 CharArrayBuffer	924
6.164.1.4 CharArrayBuffer	925
6.164.1.5 ~CharArrayBuffer	925
6.164.2 Member Function Documentation	925
6.164.2.1 array	925
6.164.2.2 arrayOffset	925
6.164.2.3 asReadOnlyBuffer	926
6.164.2.4 compact	926
6.164.2.5 duplicate	926
6.164.2.6 get	927
6.164.2.7 get	927
6.164.2.8 hasArray	927
6.164.2.9 isReadOnly	928
6.164.2.10put	928
6.164.2.11put	928
6.164.2.12setReadOnly	929
6.164.2.13slice	929
6.164.2.14subSequence	929
6.164.3Field Documentation	930
6.164.3.1 _array	930
6.164.3.2 offset	930
6.164.3.3 readOnly	930

6.165decaf::nio::CharBuffer Class Reference	930
6.165.1 Detailed Description	933
6.165.2 Constructor & Destructor Documentation	933
6.165.2.1 CharBuffer	933
6.165.2.2 ~CharBuffer	934
6.165.3 Member Function Documentation	934
6.165.3.1 allocate	934
6.165.3.2 append	934
6.165.3.3 append	935
6.165.3.4 append	935
6.165.3.5 array	935
6.165.3.6 arrayOffset	936
6.165.3.7 asReadOnlyBuffer	936
6.165.3.8 charAt	936
6.165.3.9 compact	937
6.165.3.10compareTo	937
6.165.3.11duplicate	938
6.165.3.12equals	938
6.165.3.13get	938
6.165.3.14get	938
6.165.3.15get	939
6.165.3.16get	939
6.165.3.17hasArray	940
6.165.3.18length	940
6.165.3.19operator<	940
6.165.3.20operator==	940
6.165.3.21put	941
6.165.3.22put	941
6.165.3.23put	942
6.165.3.24put	942
6.165.3.25put	942
6.165.3.26put	943
6.165.3.27put	944
6.165.3.28read	944
6.165.3.29slice	944
6.165.3.30subSequence	945

6.165.3.31toString	945
6.165.3.32wrap	945
6.165.3.33wrap	946
6.166decaf::lang::CharSequence Class Reference	946
6.166.1 Detailed Description	947
6.166.2 Constructor & Destructor Documentation	947
6.166.2.1 ~CharSequence	947
6.166.3 Member Function Documentation	947
6.166.3.1 charAt	947
6.166.3.2 length	947
6.166.3.3 subSequence	947
6.166.3.4 toString	948
6.167decaf::lang::exceptions::ClassCastException Class Reference	948
6.167.1 Constructor & Destructor Documentation	949
6.167.1.1 ClassCastException	949
6.167.1.2 ClassCastException	949
6.167.1.3 ClassCastException	949
6.167.1.4 ClassCastException	949
6.167.1.5 ClassCastException	949
6.167.1.6 ClassCastException	950
6.167.1.7 ~ClassCastException	950
6.167.2 Member Function Documentation	950
6.167.2.1 clone	950
6.168decaf::io::Closeable Class Reference	950
6.168.1 Detailed Description	951
6.168.2 Constructor & Destructor Documentation	951
6.168.2.1 ~Closeable	951
6.168.3 Member Function Documentation	951
6.168.3.1 close	951
6.169cms::Closeable Class Reference	951
6.169.1 Detailed Description	952
6.169.2 Constructor & Destructor Documentation	952
6.169.2.1 ~Closeable	952
6.169.3 Member Function Documentation	952
6.169.3.1 close	952
6.170activemq::transport::failover::CloseTransportsTask Class Reference	952

6.170.1 Constructor & Destructor Documentation	953
6.170.1.1 CloseTransportsTask	953
6.170.1.2 ~CloseTransportsTask	953
6.170.2 Member Function Documentation	953
6.170.2.1 add	953
6.170.2.2 isPending	953
6.170.2.3 iterate	953
6.171activemq::cmsutil::CmsAccessor Class Reference	954
6.171.1 Detailed Description	955
6.171.2 Constructor & Destructor Documentation	955
6.171.2.1 CmsAccessor	955
6.171.2.2 ~CmsAccessor	955
6.171.3 Member Function Documentation	955
6.171.3.1 checkConnectionFactory	955
6.171.3.2 createConnection	955
6.171.3.3 createSession	956
6.171.3.4 destroy	956
6.171.3.5 getConnectionFactory	956
6.171.3.6 getConnectionFactory	956
6.171.3.7 getResourceLifecycleManager	956
6.171.3.8 getResourceLifecycleManager	956
6.171.3.9 getSessionAcknowledgeMode	956
6.171.3.10init	957
6.171.3.11setConnectionFactory	957
6.171.3.12setSessionAcknowledgeMode	957
6.172activemq::cmsutil::CmsDestinationAccessor Class Reference	957
6.172.1 Detailed Description	958
6.172.2 Constructor & Destructor Documentation	959
6.172.2.1 CmsDestinationAccessor	959
6.172.2.2 ~CmsDestinationAccessor	959
6.172.3 Member Function Documentation	959
6.172.3.1 checkDestinationResolver	959
6.172.3.2 destroy	959
6.172.3.3 getDestinationResolver	959
6.172.3.4 getDestinationResolver	959
6.172.3.5 init	959

6.172.3.6 isPubSubDomain	960
6.172.3.7 resolveDestinationName	960
6.172.3.8 setDestinationResolver	960
6.172.3.9 setPubSubDomain	960
6.173cms::CMSException Class Reference	960
6.173.1 Detailed Description	961
6.173.2 Constructor & Destructor Documentation	962
6.173.2.1 CMSException	962
6.173.2.2 CMSException	962
6.173.2.3 CMSException	962
6.173.2.4 CMSException	962
6.173.2.5 ~CMSException	962
6.173.3 Member Function Documentation	962
6.173.3.1 getCause	962
6.173.3.2 getMessage	962
6.173.3.3 getStackTrace	962
6.173.3.4 getStackTraceString	963
6.173.3.5 printStackTrace	963
6.173.3.6 printStackTrace	963
6.173.3.7 setMark	963
6.173.3.8 what	963
6.174activemq::util::CMSExceptionSupport Class Reference	963
6.174.1 Constructor & Destructor Documentation	964
6.174.1.1 ~CMSExceptionSupport	964
6.174.2 Member Function Documentation	964
6.174.2.1 create	964
6.174.2.2 create	964
6.174.2.3 createMessageEOFException	964
6.174.2.4 createMessageFormatException	964
6.175cms::CMSProperties Class Reference	965
6.175.1 Detailed Description	966
6.175.2 Constructor & Destructor Documentation	966
6.175.2.1 ~CMSProperties	966
6.175.3 Member Function Documentation	966
6.175.3.1 clear	966
6.175.3.2 clone	966

6.175.3.3 copy	966
6.175.3.4 getProperty	966
6.175.3.5 getProperty	967
6.175.3.6 hasProperty	967
6.175.3.7 isEmpty	967
6.175.3.8 remove	967
6.175.3.9 setProperty	968
6.175.3.10oArray	968
6.175.3.11toString	968
6.176cms::CMSSecurityException Class Reference	968
6.176.1 Detailed Description	969
6.176.2 Constructor & Destructor Documentation	969
6.176.2.1 CMSSecurityException	969
6.176.2.2 CMSSecurityException	969
6.176.2.3 CMSSecurityException	969
6.176.2.4 CMSSecurityException	969
6.176.2.5 ~CMSSecurityException	969
6.177activemq::cmsutil::CmsTemplate Class Reference	969
6.177.1 Detailed Description	973
6.177.2 Constructor & Destructor Documentation	973
6.177.2.1 CmsTemplate	973
6.177.2.2 CmsTemplate	973
6.177.2.3 ~CmsTemplate	973
6.177.3 Member Function Documentation	973
6.177.3.1 destroy	973
6.177.3.2 execute	973
6.177.3.3 execute	974
6.177.3.4 execute	974
6.177.3.5 execute	974
6.177.3.6 getDefaultDestination	974
6.177.3.7 getDefaultDestination	975
6.177.3.8 getDefaultDestinationName	975
6.177.3.9 getDeliveryMode	975
6.177.3.10getPriority	975
6.177.3.11getReceiveTimeout	975
6.177.3.12getTimeToLive	975

6.177.3.13	nit	975
6.177.3.14	sExplicitQosEnabled	976
6.177.3.15	sMessageIdEnabled	976
6.177.3.16	sMessageTimestampEnabled	976
6.177.3.17	sNoLocal	976
6.177.3.18	receive	976
6.177.3.19	receive	976
6.177.3.20	receive	977
6.177.3.21	receiveSelected	977
6.177.3.22	receiveSelected	977
6.177.3.23	receiveSelected	978
6.177.3.24	end	978
6.177.3.25	end	978
6.177.3.26	end	979
6.177.3.27	setDefaultDestination	979
6.177.3.28	setDefaultDestinationName	979
6.177.3.29	setDeliveryMode	979
6.177.3.30	setDeliveryPersistent	980
6.177.3.31	setExplicitQosEnabled	980
6.177.3.32	setMessageIdEnabled	980
6.177.3.33	setMessageTimestampEnabled	980
6.177.3.34	setNoLocal	980
6.177.3.35	setPriority	980
6.177.3.36	setPubSubDomain	980
6.177.3.37	setReceiveTimeout	981
6.177.3.38	setTimeToLive	981
6.177.4	Friends And Related Function Documentation	981
6.177.4.1	ProducerExecutor	981
6.177.4.2	ReceiveExecutor	981
6.177.4.3	ResolveProducerExecutor	981
6.177.4.4	ResolveReceiveExecutor	981
6.177.4.5	SendExecutor	981
6.177.5	Field Documentation	981
6.177.5.1	DEFAULT_PRIORITY	981
6.177.5.2	DEFAULT_TIME_TO_LIVE	981
6.177.5.3	RECEIVE_TIMEOUT_INDEFINITE_WAIT	982

6.177.5.4	RECEIVE_TIMEOUT_NO_WAIT	982
6.178	decaf::util::Collection< E > Class Template Reference	982
6.178.1	Detailed Description	983
6.178.2	Constructor & Destructor Documentation	984
6.178.2.1	~Collection	984
6.178.3	Member Function Documentation	984
6.178.3.1	add	984
6.178.3.2	addAll	985
6.178.3.3	clear	985
6.178.3.4	contains	986
6.178.3.5	containsAll	986
6.178.3.6	equals	987
6.178.3.7	isEmpty	987
6.178.3.8	remove	988
6.178.3.9	removeAll	988
6.178.3.10	retainAll	989
6.178.3.11	size	990
6.178.3.12	toArray	990
6.179	activemq::commands::Command Class Reference	991
6.179.1	Constructor & Destructor Documentation	992
6.179.1.1	~Command	992
6.179.2	Member Function Documentation	992
6.179.2.1	getCommandId	992
6.179.2.2	isBrokerInfo	992
6.179.2.3	isConnectionInfo	992
6.179.2.4	isConsumerInfo	992
6.179.2.5	isKeepAliveInfo	992
6.179.2.6	isMessage	993
6.179.2.7	isMessageAck	993
6.179.2.8	isMessageDispatch	993
6.179.2.9	isMessageDispatchNotification	993
6.179.2.10	isProducerAck	993
6.179.2.11	isProducerInfo	993
6.179.2.12	isRemoveInfo	993
6.179.2.13	isRemoveSubscriptionInfo	993
6.179.2.14	isResponse	994

6.179.2.15sResponseRequired	994
6.179.2.16sShutdownInfo	994
6.179.2.17sTransactionInfo	994
6.179.2.18sWireFormatInfo	994
6.179.2.19set CommandId	994
6.179.2.20set ResponseRequired	994
6.179.2.21toString	995
6.179.2.22visit	995
6.180activemq::state::CommandVisitor Class Reference	996
6.180.1 Detailed Description	998
6.180.2 Constructor & Destructor Documentation	998
6.180.2.1 ~CommandVisitor	998
6.180.3 Member Function Documentation	998
6.180.3.1 processBeginTransaction	998
6.180.3.2 processBrokerError	999
6.180.3.3 processBrokerInfo	999
6.180.3.4 processCommitTransactionOnePhase	999
6.180.3.5 processCommitTransactionTwoPhase	999
6.180.3.6 processConnectionControl	999
6.180.3.7 processConnectionError	999
6.180.3.8 processConnectionInfo	999
6.180.3.9 processConsumerControl	999
6.180.3.10processConsumerInfo	999
6.180.3.11processControlCommand	1000
6.180.3.12processDestinationInfo	1000
6.180.3.13processEndTransaction	1000
6.180.3.14processFlushCommand	1000
6.180.3.15processForgetTransaction	1000
6.180.3.16processKeepAliveInfo	1000
6.180.3.17processMessage	1000
6.180.3.18processMessageAck	1000
6.180.3.19processMessageDispatch	1001
6.180.3.20processMessageDispatchNotification	1001
6.180.3.21processMessagePull	1001
6.180.3.22processPrepareTransaction	1001
6.180.3.23processProducerAck	1001

6.180.3.24	processProducerInfo	1001
6.180.3.25	processRecoverTransactions	1001
6.180.3.26	processRemoveConnection	1001
6.180.3.27	processRemoveConsumer	1001
6.180.3.28	processRemoveDestination	1002
6.180.3.29	processRemoveInfo	1002
6.180.3.30	processRemoveProducer	1002
6.180.3.31	processRemoveSession	1002
6.180.3.32	processRemoveSubscriptionInfo	1002
6.180.3.33	processReplayCommand	1002
6.180.3.34	processResponse	1002
6.180.3.35	processRollbackTransaction	1002
6.180.3.36	processSessionInfo	1003
6.180.3.37	processShutdownInfo	1003
6.180.3.38	processTransactionInfo	1003
6.180.3.39	processWireFormat	1003
6.181	activemq::state::CommandVisitorAdapter Class Reference	1003
6.181.1	Detailed Description	1006
6.181.2	Constructor & Destructor Documentation	1007
6.181.2.1	~CommandVisitorAdapter	1007
6.181.3	Member Function Documentation	1007
6.181.3.1	processBeginTransaction	1007
6.181.3.2	processBrokerError	1007
6.181.3.3	processBrokerInfo	1007
6.181.3.4	processCommitTransactionOnePhase	1007
6.181.3.5	processCommitTransactionTwoPhase	1007
6.181.3.6	processConnectionControl	1007
6.181.3.7	processConnectionError	1007
6.181.3.8	processConnectionInfo	1007
6.181.3.9	processConsumerControl	1007
6.181.3.10	processConsumerInfo	1007
6.181.3.11	processControlCommand	1007
6.181.3.12	processDestinationInfo	1007
6.181.3.13	processEndTransaction	1007
6.181.3.14	processFlushCommand	1007
6.181.3.15	processForgetTransaction	1007

6.181.3.16	processKeepAliveInfo	1007
6.181.3.17	processMessage	1007
6.181.3.18	processMessageAck	1007
6.181.3.19	processMessageDispatch	1007
6.181.3.20	processMessageDispatchNotification	1007
6.181.3.21	processMessagePull	1007
6.181.3.22	processPrepareTransaction	1007
6.181.3.23	processProducerAck	1007
6.181.3.24	processProducerInfo	1007
6.181.3.25	processRecoverTransactions	1007
6.181.3.26	processRemoveConnection	1007
6.181.3.27	processRemoveConsumer	1007
6.181.3.28	processRemoveDestination	1007
6.181.3.29	processRemoveInfo	1007
6.181.3.30	processRemoveProducer	1008
6.181.3.31	processRemoveSession	1008
6.181.3.32	processRemoveSubscriptionInfo	1008
6.181.3.33	processReplayCommand	1008
6.181.3.34	processResponse	1008
6.181.3.35	processRollbackTransaction	1008
6.181.3.36	processSessionInfo	1008
6.181.3.37	processShutdownInfo	1008
6.181.3.38	processTransactionInfo	1008
6.181.3.39	processWireFormat	1009
6.182	decaf::lang::Comparable< T > Class Template Reference	1009
6.182.1	Detailed Description	1009
6.182.2	Constructor & Destructor Documentation	1010
6.182.2.1	~Comparable	1010
6.182.3	Member Function Documentation	1010
6.182.3.1	compareTo	1010
6.182.3.2	equals	1010
6.182.3.3	operator<	1011
6.182.3.4	operator==	1011
6.183	decaf::util::Comparator< T > Class Template Reference	1011
6.183.1	Detailed Description	1012
6.183.2	Constructor & Destructor Documentation	1012

6.183.2.1 ~Comparator	1012
6.183.3 Member Function Documentation	1012
6.183.3.1 compare	1012
6.183.3.2 operator()	1013
6.184activemq::util::CompositeData Class Reference	1013
6.184.1 Detailed Description	1014
6.184.2 Constructor & Destructor Documentation	1015
6.184.2.1 CompositeData	1015
6.184.2.2 ~CompositeData	1015
6.184.3 Member Function Documentation	1015
6.184.3.1 getComponents	1015
6.184.3.2 getComponents	1015
6.184.3.3 getFragment	1015
6.184.3.4 getHost	1015
6.184.3.5 getParameters	1015
6.184.3.6 getPath	1015
6.184.3.7 getScheme	1015
6.184.3.8 setComponents	1015
6.184.3.9 setFragment	1015
6.184.3.10setHost	1015
6.184.3.11setParameters	1015
6.184.3.12setPath	1015
6.184.3.13setScheme	1015
6.184.3.14oURI	1015
6.185activemq::threads::CompositeTask Class Reference	1016
6.185.1 Detailed Description	1016
6.185.2 Constructor & Destructor Documentation	1016
6.185.2.1 ~CompositeTask	1016
6.185.3 Member Function Documentation	1016
6.185.3.1 isPending	1016
6.186activemq::threads::CompositeTaskRunner Class Reference	1017
6.186.1 Detailed Description	1017
6.186.2 Constructor & Destructor Documentation	1018
6.186.2.1 CompositeTaskRunner	1018
6.186.2.2 ~CompositeTaskRunner	1018
6.186.3 Member Function Documentation	1018

6.186.3.1	addTask	1018
6.186.3.2	iterate	1018
6.186.3.3	removeTask	1018
6.186.3.4	run	1018
6.186.3.5	shutdown	1018
6.186.3.6	shutdown	1018
6.186.3.7	wakeup	1019
6.187	activemq::transport::CompositeTransport Class Reference	1019
6.187.1	Detailed Description	1019
6.187.2	Constructor & Destructor Documentation	1020
6.187.2.1	~CompositeTransport	1020
6.187.3	Member Function Documentation	1020
6.187.3.1	addURI	1020
6.187.3.2	removeURI	1020
6.188	decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR > Class Template Reference	1020
6.188.1	Detailed Description	1021
6.188.2	Constructor & Destructor Documentation	1021
6.188.2.1	~ConcurrentMap	1021
6.188.3	Member Function Documentation	1021
6.188.3.1	putIfAbsent	1021
6.188.3.2	remove	1022
6.188.3.3	replace	1023
6.188.3.4	replace	1024
6.189	decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR > Class Template Reference	1024
6.189.1	Detailed Description	1028
6.189.2	Constructor & Destructor Documentation	1029
6.189.2.1	ConcurrentStlMap	1029
6.189.2.2	ConcurrentStlMap	1029
6.189.2.3	ConcurrentStlMap	1029
6.189.2.4	~ConcurrentStlMap	1029
6.189.3	Member Function Documentation	1029
6.189.3.1	clear	1029
6.189.3.2	containsKey	1030
6.189.3.3	containsValue	1030
6.189.3.4	copy	1030

6.189.3.5 copy	1031
6.189.3.6 equals	1031
6.189.3.7 equals	1031
6.189.3.8 get	1031
6.189.3.9 get	1032
6.189.3.10 isEmpty	1032
6.189.3.11 keySet	1032
6.189.3.12 lock	1033
6.189.3.13 notify	1033
6.189.3.14 notifyAll	1033
6.189.3.15 put	1034
6.189.3.16 putAll	1034
6.189.3.17 putAll	1034
6.189.3.18 putIfAbsent	1035
6.189.3.19 remove	1035
6.189.3.20 remove	1036
6.189.3.21 replace	1036
6.189.3.22 replace	1037
6.189.3.23 size	1037
6.189.3.24 tryLock	1038
6.189.3.25 unlock	1038
6.189.3.26 values	1038
6.189.3.27 wait	1038
6.189.3.28 wait	1039
6.189.3.29 wait	1039
6.190 decaf::util::concurrent::locks::Condition Class Reference	1040
6.190.1 Detailed Description	1041
6.190.2 Constructor & Destructor Documentation	1042
6.190.2.1 ~Condition	1042
6.190.3 Member Function Documentation	1042
6.190.3.1 await	1042
6.190.3.2 await	1043
6.190.3.3 awaitNanos	1044
6.190.3.4 awaitUninterruptibly	1045
6.190.3.5 awaitUntil	1046
6.190.3.6 signal	1046

6.190.3.7 signalAll	1046
6.191decaf::util::concurrent::ConditionHandle Class Reference	1046
6.191.1 Constructor & Destructor Documentation	1047
6.191.1.1 ConditionHandle	1047
6.191.1.2 ~ConditionHandle	1047
6.191.1.3 ConditionHandle	1047
6.191.1.4 ~ConditionHandle	1047
6.191.2 Field Documentation	1047
6.191.2.1 condition	1047
6.191.2.2 criticalSection	1047
6.191.2.3 generation	1047
6.191.2.4 mutex	1047
6.191.2.5 numWaiting	1047
6.191.2.6 numWake	1047
6.191.2.7 semaphore	1047
6.192decaf::internal::util::concurrent::ConditionImpl Class Reference	1047
6.192.1 Member Function Documentation	1048
6.192.1.1 create	1048
6.192.1.2 destroy	1048
6.192.1.3 notify	1049
6.192.1.4 notifyAll	1049
6.192.1.5 wait	1049
6.192.1.6 wait	1049
6.193decaf::net::ConnectException Class Reference	1049
6.193.1 Constructor & Destructor Documentation	1050
6.193.1.1 ConnectException	1050
6.193.1.2 ConnectException	1050
6.193.1.3 ConnectException	1050
6.193.1.4 ConnectException	1051
6.193.1.5 ConnectException	1051
6.193.1.6 ConnectException	1051
6.193.1.7 ~ConnectException	1051
6.193.2 Member Function Documentation	1051
6.193.2.1 clone	1051
6.194cms::Connection Class Reference	1052
6.194.1 Detailed Description	1053

6.194.2 Constructor & Destructor Documentation	1053
6.194.2.1 ~Connection	1053
6.194.3 Member Function Documentation	1053
6.194.3.1 close	1053
6.194.3.2 createSession	1054
6.194.3.3 createSession	1054
6.194.3.4 getClientID	1054
6.194.3.5 getExceptionListener	1054
6.194.3.6 getMetaData	1054
6.194.3.7 setExceptionListener	1055
6.195activemq::commands::ConnectionControl Class Reference	1055
6.195.1 Constructor & Destructor Documentation	1057
6.195.1.1 ConnectionControl	1057
6.195.1.2 ConnectionControl	1057
6.195.1.3 ~ConnectionControl	1057
6.195.2 Member Function Documentation	1057
6.195.2.1 cloneDataStructure	1057
6.195.2.2 copyDataStructure	1057
6.195.2.3 equals	1057
6.195.2.4 getDataStructureType	1057
6.195.2.5 isClose	1058
6.195.2.6 isExit	1058
6.195.2.7 isFaultTolerant	1058
6.195.2.8 isResume	1058
6.195.2.9 isSuspend	1058
6.195.2.10operator=	1058
6.195.2.11setClose	1058
6.195.2.12setExit	1058
6.195.2.13setFaultTolerant	1058
6.195.2.14setResume	1058
6.195.2.15setSuspend	1058
6.195.2.16toString	1058
6.195.2.17visit	1059
6.195.3 Field Documentation	1059
6.195.3.1 close	1059
6.195.3.2 exit	1059

6.195.3.3	faultTolerant	1059
6.195.3.4	ID_CONNECTIONCONTROL	1059
6.195.3.5	resume	1059
6.195.3.6	suspend	1059
6.196	activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller Class	
	Reference	1059
6.196.1	Detailed Description	1060
6.196.2	Constructor & Destructor Documentation	1061
6.196.2.1	ConnectionControlMarshaller	1061
6.196.2.2	~ConnectionControlMarshaller	1061
6.196.3	Member Function Documentation	1061
6.196.3.1	createObject	1061
6.196.3.2	getDataStructureType	1061
6.196.3.3	looseMarshal	1061
6.196.3.4	looseUnmarshal	1062
6.196.3.5	tightMarshal1	1062
6.196.3.6	tightMarshal2	1062
6.196.3.7	tightUnmarshal	1063
6.197	activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller Class	
	Reference	1063
6.197.1	Detailed Description	1064
6.197.2	Constructor & Destructor Documentation	1065
6.197.2.1	ConnectionControlMarshaller	1065
6.197.2.2	~ConnectionControlMarshaller	1065
6.197.3	Member Function Documentation	1065
6.197.3.1	createObject	1065
6.197.3.2	getDataStructureType	1065
6.197.3.3	looseMarshal	1065
6.197.3.4	looseUnmarshal	1066
6.197.3.5	tightMarshal1	1066
6.197.3.6	tightMarshal2	1066
6.197.3.7	tightUnmarshal	1067
6.198	activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller Class	
	Reference	1067
6.198.1	Detailed Description	1068
6.198.2	Constructor & Destructor Documentation	1069
6.198.2.1	ConnectionControlMarshaller	1069

6.198.2.2 ~ConnectionControlMarshaller	1069
6.198.3 Member Function Documentation	1069
6.198.3.1 createObject	1069
6.198.3.2 getDataStructureType	1069
6.198.3.3 looseMarshal	1069
6.198.3.4 looseUnmarshal	1070
6.198.3.5 tightMarshal1	1070
6.198.3.6 tightMarshal2	1070
6.198.3.7 tightUnmarshal	1071
6.199activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller Class	
Reference	1071
6.199.1 Detailed Description	1072
6.199.2 Constructor & Destructor Documentation	1073
6.199.2.1 ConnectionControlMarshaller	1073
6.199.2.2 ~ConnectionControlMarshaller	1073
6.199.3 Member Function Documentation	1073
6.199.3.1 createObject	1073
6.199.3.2 getDataStructureType	1073
6.199.3.3 looseMarshal	1073
6.199.3.4 looseUnmarshal	1074
6.199.3.5 tightMarshal1	1074
6.199.3.6 tightMarshal2	1074
6.199.3.7 tightUnmarshal	1075
6.200activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller Class	
Reference	1075
6.200.1 Detailed Description	1076
6.200.2 Constructor & Destructor Documentation	1077
6.200.2.1 ConnectionControlMarshaller	1077
6.200.2.2 ~ConnectionControlMarshaller	1077
6.200.3 Member Function Documentation	1077
6.200.3.1 createObject	1077
6.200.3.2 getDataStructureType	1077
6.200.3.3 looseMarshal	1077
6.200.3.4 looseUnmarshal	1078
6.200.3.5 tightMarshal1	1078
6.200.3.6 tightMarshal2	1078
6.200.3.7 tightUnmarshal	1079

6.201activemq::commands::ConnectionError Class Reference	1079
6.201.1 Constructor & Destructor Documentation	1081
6.201.1.1 ConnectionError	1081
6.201.1.2 ConnectionError	1081
6.201.1.3 ~ConnectionError	1081
6.201.2 Member Function Documentation	1081
6.201.2.1 cloneDataStructure	1081
6.201.2.2 copyDataStructure	1081
6.201.2.3 equals	1081
6.201.2.4 getConnectionId	1082
6.201.2.5 getConnectionId	1082
6.201.2.6 getDataStructureType	1082
6.201.2.7 getException	1082
6.201.2.8 getException	1082
6.201.2.9 operator=	1082
6.201.2.10 setConnectionId	1082
6.201.2.11 setException	1082
6.201.2.12 toString	1082
6.201.2.13 visit	1083
6.201.3 Field Documentation	1083
6.201.3.1 connectionId	1083
6.201.3.2 exception	1083
6.201.3.3 ID_CONNECTIONERROR	1083
6.202activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller Class Reference	1083
6.202.1 Detailed Description	1084
6.202.2 Constructor & Destructor Documentation	1084
6.202.2.1 ConnectionErrorMarshaller	1084
6.202.2.2 ~ConnectionErrorMarshaller	1084
6.202.3 Member Function Documentation	1084
6.202.3.1 createObject	1084
6.202.3.2 getDataStructureType	1085
6.202.3.3 looseMarshal	1085
6.202.3.4 looseUnmarshal	1085
6.202.3.5 tightMarshal1	1086
6.202.3.6 tightMarshal2	1086
6.202.3.7 tightUnmarshal	1087

6.203	activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller	Class	
	Reference		1087
6.203.1	Detailed Description		1088
6.203.2	Constructor & Destructor Documentation		1088
6.203.2.1	ConnectionErrorMarshaller		1088
6.203.2.2	~ConnectionErrorMarshaller		1088
6.203.3	Member Function Documentation		1088
6.203.3.1	createObject		1088
6.203.3.2	getDataStructureType		1089
6.203.3.3	looseMarshal		1089
6.203.3.4	looseUnmarshal		1089
6.203.3.5	tightMarshal1		1090
6.203.3.6	tightMarshal2		1090
6.203.3.7	tightUnmarshal		1090
6.204	activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller	Class	
	Reference		1091
6.204.1	Detailed Description		1092
6.204.2	Constructor & Destructor Documentation		1092
6.204.2.1	ConnectionErrorMarshaller		1092
6.204.2.2	~ConnectionErrorMarshaller		1092
6.204.3	Member Function Documentation		1092
6.204.3.1	createObject		1092
6.204.3.2	getDataStructureType		1092
6.204.3.3	looseMarshal		1093
6.204.3.4	looseUnmarshal		1093
6.204.3.5	tightMarshal1		1093
6.204.3.6	tightMarshal2		1094
6.204.3.7	tightUnmarshal		1094
6.205	activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller	Class	
	Reference		1095
6.205.1	Detailed Description		1096
6.205.2	Constructor & Destructor Documentation		1096
6.205.2.1	ConnectionErrorMarshaller		1096
6.205.2.2	~ConnectionErrorMarshaller		1096
6.205.3	Member Function Documentation		1096
6.205.3.1	createObject		1096
6.205.3.2	getDataStructureType		1096

6.205.3.3	looseMarshal	1097
6.205.3.4	looseUnmarshal	1097
6.205.3.5	tightMarshal1	1097
6.205.3.6	tightMarshal2	1098
6.205.3.7	tightUnmarshal	1098
6.206	activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller Class Reference	1099
6.206.1	Detailed Description	1100
6.206.2	Constructor & Destructor Documentation	1100
6.206.2.1	ConnectionErrorMarshaller	1100
6.206.2.2	~ConnectionErrorMarshaller	1100
6.206.3	Member Function Documentation	1100
6.206.3.1	createObject	1100
6.206.3.2	getDataStructureType	1100
6.206.3.3	looseMarshal	1101
6.206.3.4	looseUnmarshal	1101
6.206.3.5	tightMarshal1	1101
6.206.3.6	tightMarshal2	1102
6.206.3.7	tightUnmarshal	1102
6.207	cms::ConnectionFactory Class Reference	1103
6.207.1	Detailed Description	1103
6.207.2	Constructor & Destructor Documentation	1104
6.207.2.1	~ConnectionFactory	1104
6.207.3	Member Function Documentation	1104
6.207.3.1	createCMSConnectionFactory	1104
6.207.3.2	createConnection	1104
6.207.3.3	createConnection	1105
6.207.3.4	createConnection	1105
6.208	activemq::commands::ConnectionId Class Reference	1106
6.208.1	Member Typedef Documentation	1107
6.208.1.1	COMPARATOR	1107
6.208.2	Constructor & Destructor Documentation	1107
6.208.2.1	ConnectionId	1107
6.208.2.2	ConnectionId	1107
6.208.2.3	ConnectionId	1107
6.208.2.4	ConnectionId	1107
6.208.2.5	ConnectionId	1107

6.208.2.6 ~ConnectionId	1107
6.208.3 Member Function Documentation	1107
6.208.3.1 cloneDataStructure	1107
6.208.3.2 compareTo	1107
6.208.3.3 copyDataStructure	1107
6.208.3.4 equals	1108
6.208.3.5 equals	1108
6.208.3.6 getDataStructureType	1108
6.208.3.7 getValue	1108
6.208.3.8 getValue	1108
6.208.3.9 operator<	1108
6.208.3.10 operator=	1108
6.208.3.11 operator==	1108
6.208.3.12 setValue	1108
6.208.3.13 toString	1108
6.208.4 Field Documentation	1109
6.208.4.1 ID_CONNECTIONID	1109
6.208.4.2 value	1109
6.209activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller Class Reference	1109
6.209.1 Detailed Description	1110
6.209.2 Constructor & Destructor Documentation	1110
6.209.2.1 ConnectionIdMarshaller	1110
6.209.2.2 ~ConnectionIdMarshaller	1110
6.209.3 Member Function Documentation	1110
6.209.3.1 createObject	1110
6.209.3.2 getDataStructureType	1110
6.209.3.3 looseMarshal	1111
6.209.3.4 looseUnmarshal	1111
6.209.3.5 tightMarshal1	1111
6.209.3.6 tightMarshal2	1112
6.209.3.7 tightUnmarshal	1112
6.210activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller Class Reference	1113
6.210.1 Detailed Description	1114
6.210.2 Constructor & Destructor Documentation	1114
6.210.2.1 ConnectionIdMarshaller	1114

6.210.2.2	~ConnectionIdMarshaller	1114
6.210.3	Member Function Documentation	1114
6.210.3.1	createObject	1114
6.210.3.2	getDataStructureType	1114
6.210.3.3	looseMarshal	1114
6.210.3.4	looseUnmarshal	1115
6.210.3.5	tightMarshal1	1115
6.210.3.6	tightMarshal2	1116
6.210.3.7	tightUnmarshal	1116
6.211	activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller Class Reference	1116
6.211.1	Detailed Description	1117
6.211.2	Constructor & Destructor Documentation	1118
6.211.2.1	ConnectionIdMarshaller	1118
6.211.2.2	~ConnectionIdMarshaller	1118
6.211.3	Member Function Documentation	1118
6.211.3.1	createObject	1118
6.211.3.2	getDataStructureType	1118
6.211.3.3	looseMarshal	1118
6.211.3.4	looseUnmarshal	1119
6.211.3.5	tightMarshal1	1119
6.211.3.6	tightMarshal2	1119
6.211.3.7	tightUnmarshal	1120
6.212	activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller Class Reference	1120
6.212.1	Detailed Description	1121
6.212.2	Constructor & Destructor Documentation	1122
6.212.2.1	ConnectionIdMarshaller	1122
6.212.2.2	~ConnectionIdMarshaller	1122
6.212.3	Member Function Documentation	1122
6.212.3.1	createObject	1122
6.212.3.2	getDataStructureType	1122
6.212.3.3	looseMarshal	1122
6.212.3.4	looseUnmarshal	1123
6.212.3.5	tightMarshal1	1123
6.212.3.6	tightMarshal2	1123
6.212.3.7	tightUnmarshal	1124

6.213activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller Class Reference	1124
6.213.1 Detailed Description	1125
6.213.2 Constructor & Destructor Documentation	1126
6.213.2.1 ConnectionIdMarshaller	1126
6.213.2.2 ~ConnectionIdMarshaller	1126
6.213.3 Member Function Documentation	1126
6.213.3.1 createObject	1126
6.213.3.2 getDataStructureType	1126
6.213.3.3 looseMarshal	1126
6.213.3.4 looseUnmarshal	1127
6.213.3.5 tightMarshal1	1127
6.213.3.6 tightMarshal2	1127
6.213.3.7 tightUnmarshal	1128
6.214activemq::commands::ConnectionInfo Class Reference	1128
6.214.1 Constructor & Destructor Documentation	1130
6.214.1.1 ConnectionInfo	1130
6.214.1.2 ConnectionInfo	1130
6.214.1.3 ~ConnectionInfo	1130
6.214.2 Member Function Documentation	1130
6.214.2.1 cloneDataStructure	1130
6.214.2.2 copyDataStructure	1130
6.214.2.3 equals	1131
6.214.2.4 getBrokerPath	1131
6.214.2.5 getBrokerPath	1131
6.214.2.6 getClientId	1131
6.214.2.7 getClientId	1131
6.214.2.8 getConnectionId	1131
6.214.2.9 getConnectionId	1131
6.214.2.10getDataStructureType	1131
6.214.2.11getPassword	1132
6.214.2.12getPassword	1132
6.214.2.13getUserName	1132
6.214.2.14getUserName	1132
6.214.2.15sBrokerMasterConnector	1132
6.214.2.16sClientMaster	1132
6.214.2.17sConnectionInfo	1132

6.214.2.18	manageable	1133
6.214.2.19	operator=	1133
6.214.2.20	setBrokerMasterConnector	1133
6.214.2.21	setBrokerPath	1133
6.214.2.22	setClientId	1133
6.214.2.23	setClientMaster	1133
6.214.2.24	setConnectionId	1133
6.214.2.25	setManageable	1133
6.214.2.26	setPassword	1133
6.214.2.27	setUserName	1133
6.214.2.28	toString	1133
6.214.2.29	visit	1133
6.214.3	Field Documentation	1134
6.214.3.1	brokerMasterConnector	1134
6.214.3.2	brokerPath	1134
6.214.3.3	clientId	1134
6.214.3.4	clientMaster	1134
6.214.3.5	connectionId	1134
6.214.3.6	ID_CONNECTIONINFO	1134
6.214.3.7	manageable	1134
6.214.3.8	password	1134
6.214.3.9	userName	1134
6.215	activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller Class Reference	1134
6.215.1	Detailed Description	1135
6.215.2	Constructor & Destructor Documentation	1136
6.215.2.1	ConnectionInfoMarshaller	1136
6.215.2.2	~ConnectionInfoMarshaller	1136
6.215.3	Member Function Documentation	1136
6.215.3.1	createObject	1136
6.215.3.2	getDataStructureType	1136
6.215.3.3	looseMarshal	1136
6.215.3.4	looseUnmarshal	1137
6.215.3.5	tightMarshal1	1137
6.215.3.6	tightMarshal2	1137
6.215.3.7	tightUnmarshal	1138

6.216activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller Class Reference	1138
6.216.1 Detailed Description	1139
6.216.2 Constructor & Destructor Documentation	1140
6.216.2.1 ConnectionInfoMarshaller	1140
6.216.2.2 ~ConnectionInfoMarshaller	1140
6.216.3 Member Function Documentation	1140
6.216.3.1 createObject	1140
6.216.3.2 getDataStructureType	1140
6.216.3.3 looseMarshal	1140
6.216.3.4 looseUnmarshal	1141
6.216.3.5 tightMarshal1	1141
6.216.3.6 tightMarshal2	1141
6.216.3.7 tightUnmarshal	1142
6.217activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller Class Reference	1142
6.217.1 Detailed Description	1143
6.217.2 Constructor & Destructor Documentation	1144
6.217.2.1 ConnectionInfoMarshaller	1144
6.217.2.2 ~ConnectionInfoMarshaller	1144
6.217.3 Member Function Documentation	1144
6.217.3.1 createObject	1144
6.217.3.2 getDataStructureType	1144
6.217.3.3 looseMarshal	1144
6.217.3.4 looseUnmarshal	1145
6.217.3.5 tightMarshal1	1145
6.217.3.6 tightMarshal2	1145
6.217.3.7 tightUnmarshal	1146
6.218activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller Class Reference	1146
6.218.1 Detailed Description	1147
6.218.2 Constructor & Destructor Documentation	1148
6.218.2.1 ConnectionInfoMarshaller	1148
6.218.2.2 ~ConnectionInfoMarshaller	1148
6.218.3 Member Function Documentation	1148
6.218.3.1 createObject	1148
6.218.3.2 getDataStructureType	1148

6.218.3.3 looseMarshal	1148
6.218.3.4 looseUnmarshal	1149
6.218.3.5 tightMarshal1	1149
6.218.3.6 tightMarshal2	1149
6.218.3.7 tightUnmarshal	1150
6.219activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller Class Reference	1150
6.219.1 Detailed Description	1151
6.219.2 Constructor & Destructor Documentation	1152
6.219.2.1 ConnectionInfoMarshaller	1152
6.219.2.2 ~ConnectionInfoMarshaller	1152
6.219.3 Member Function Documentation	1152
6.219.3.1 createObject	1152
6.219.3.2 getDataStructureType	1152
6.219.3.3 looseMarshal	1152
6.219.3.4 looseUnmarshal	1153
6.219.3.5 tightMarshal1	1153
6.219.3.6 tightMarshal2	1153
6.219.3.7 tightUnmarshal	1154
6.220cms::ConnectionMetaData Class Reference	1154
6.220.1 Detailed Description	1155
6.220.2 Constructor & Destructor Documentation	1155
6.220.2.1 ~ConnectionMetaData	1155
6.220.3 Member Function Documentation	1155
6.220.3.1 getCMSMajorVersion	1155
6.220.3.2 getCMSMinorVersion	1156
6.220.3.3 getCMSProviderName	1156
6.220.3.4 getCMSVersion	1156
6.220.3.5 getCMSXPropertyNames	1157
6.220.3.6 getProviderMajorVersion	1157
6.220.3.7 getProviderMinorVersion	1157
6.220.3.8 getProviderVersion	1157
6.221activemq::state::ConnectionState Class Reference	1158
6.221.1 Constructor & Destructor Documentation	1159
6.221.1.1 ConnectionState	1159
6.221.1.2 ~ConnectionState	1159
6.221.2 Member Function Documentation	1159

6.221.2.1 addSession	1159
6.221.2.2 addTempDestination	1159
6.221.2.3 addTransactionState	1159
6.221.2.4 checkShutdown	1159
6.221.2.5 getInfo	1159
6.221.2.6 getSessionState	1159
6.221.2.7 getSessionStates	1159
6.221.2.8 getTempDesinations	1159
6.221.2.9 getTransactionState	1159
6.221.2.10getTransactionStates	1159
6.221.2.11removeSession	1159
6.221.2.12removeTempDestination	1159
6.221.2.13removeTransactionState	1160
6.221.2.14reset	1160
6.221.2.15shutdown	1160
6.221.2.16oString	1160
6.222activemq::state::ConnectionStateTracker Class Reference	1160
6.222.1 Constructor & Destructor Documentation	1162
6.222.1.1 ConnectionStateTracker	1162
6.222.1.2 ~ConnectionStateTracker	1162
6.222.2 Member Function Documentation	1162
6.222.2.1 getMaxCacheSize	1162
6.222.2.2 isRestoreConsumers	1162
6.222.2.3 isRestoreProducers	1162
6.222.2.4 isRestoreSessions	1162
6.222.2.5 isRestoreTransaction	1162
6.222.2.6 isTrackMessages	1162
6.222.2.7 isTrackTransactions	1162
6.222.2.8 processBeginTransaction	1162
6.222.2.9 processCommitTransactionOnePhase	1162
6.222.2.10processCommitTransactionTwoPhase	1162
6.222.2.11processConnectionInfo	1163
6.222.2.12processConsumerInfo	1163
6.222.2.13processDestinationInfo	1163
6.222.2.14processEndTransaction	1163
6.222.2.15processMessage	1163

6.222.2.16	processMessageAck	1163
6.222.2.17	processPrepareTransaction	1163
6.222.2.18	processProducerInfo	1164
6.222.2.19	processRemoveConnection	1164
6.222.2.20	processRemoveConsumer	1164
6.222.2.21	processRemoveDestination	1164
6.222.2.22	processRemoveProducer	1164
6.222.2.23	processRemoveSession	1164
6.222.2.24	processRollbackTransaction	1164
6.222.2.25	processSessionInfo	1165
6.222.2.26	restore	1165
6.222.2.27	setMaxCacheSize	1165
6.222.2.28	setRestoreConsumers	1165
6.222.2.29	setRestoreProducers	1165
6.222.2.30	setRestoreSessions	1165
6.222.2.31	setRestoreTransaction	1165
6.222.2.32	setTrackMessages	1165
6.222.2.33	setTrackTransactions	1165
6.222.2.34	track	1165
6.222.2.35	trackBack	1165
6.222.3	Friends And Related Function Documentation	1165
6.222.3.1	RemoveTransactionAction	1165
6.223	activemq::commands::ConsumerControl Class Reference	1165
6.223.1	Constructor & Destructor Documentation	1167
6.223.1.1	ConsumerControl	1167
6.223.1.2	ConsumerControl	1167
6.223.1.3	~ConsumerControl	1167
6.223.2	Member Function Documentation	1167
6.223.2.1	cloneDataStructure	1167
6.223.2.2	copyDataStructure	1167
6.223.2.3	equals	1168
6.223.2.4	getConsumerId	1168
6.223.2.5	getConsumerId	1168
6.223.2.6	getDataStructureType	1168
6.223.2.7	getPrefetch	1169
6.223.2.8	isClose	1169

6.223.2.9 isFlush	1169
6.223.2.10 isStart	1169
6.223.2.11 isStop	1169
6.223.2.12 operator=	1169
6.223.2.13 setClose	1169
6.223.2.14 setConsumerId	1169
6.223.2.15 setFlush	1169
6.223.2.16 setPrefetch	1169
6.223.2.17 setStart	1169
6.223.2.18 setStop	1169
6.223.2.19 toString	1169
6.223.2.20 visit	1170
6.223.3 Field Documentation	1170
6.223.3.1 close	1170
6.223.3.2 consumerId	1170
6.223.3.3 flush	1170
6.223.3.4 ID_CONSUMERCONTROL	1170
6.223.3.5 prefetch	1170
6.223.3.6 start	1170
6.223.3.7 stop	1170
6.224activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller Class	
Reference	1170
6.224.1 Detailed Description	1171
6.224.2 Constructor & Destructor Documentation	1172
6.224.2.1 ConsumerControlMarshaller	1172
6.224.2.2 ~ConsumerControlMarshaller	1172
6.224.3 Member Function Documentation	1172
6.224.3.1 createObject	1172
6.224.3.2 getDataStructureType	1172
6.224.3.3 looseMarshal	1172
6.224.3.4 looseUnmarshal	1173
6.224.3.5 tightMarshal1	1173
6.224.3.6 tightMarshal2	1173
6.224.3.7 tightUnmarshal	1174
6.225activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller Class	
Reference	1174
6.225.1 Detailed Description	1175

6.225.2 Constructor & Destructor Documentation	1176
6.225.2.1 ConsumerControlMarshaller	1176
6.225.2.2 ~ConsumerControlMarshaller	1176
6.225.3 Member Function Documentation	1176
6.225.3.1 createObject	1176
6.225.3.2 getDataStructureType	1176
6.225.3.3 looseMarshal	1176
6.225.3.4 looseUnmarshal	1177
6.225.3.5 tightMarshal1	1177
6.225.3.6 tightMarshal2	1177
6.225.3.7 tightUnmarshal	1178
6.226activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller Class	
Reference	1178
6.226.1 Detailed Description	1179
6.226.2 Constructor & Destructor Documentation	1180
6.226.2.1 ConsumerControlMarshaller	1180
6.226.2.2 ~ConsumerControlMarshaller	1180
6.226.3 Member Function Documentation	1180
6.226.3.1 createObject	1180
6.226.3.2 getDataStructureType	1180
6.226.3.3 looseMarshal	1180
6.226.3.4 looseUnmarshal	1181
6.226.3.5 tightMarshal1	1181
6.226.3.6 tightMarshal2	1181
6.226.3.7 tightUnmarshal	1182
6.227activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller Class	
Reference	1182
6.227.1 Detailed Description	1183
6.227.2 Constructor & Destructor Documentation	1184
6.227.2.1 ConsumerControlMarshaller	1184
6.227.2.2 ~ConsumerControlMarshaller	1184
6.227.3 Member Function Documentation	1184
6.227.3.1 createObject	1184
6.227.3.2 getDataStructureType	1184
6.227.3.3 looseMarshal	1184
6.227.3.4 looseUnmarshal	1185
6.227.3.5 tightMarshal1	1185

6.227.3.6 tightMarshal2	1185
6.227.3.7 tightUnmarshal	1186
6.228activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller Class	
Reference	1186
6.228.1 Detailed Description	1187
6.228.2 Constructor & Destructor Documentation	1188
6.228.2.1 ConsumerControlMarshaller	1188
6.228.2.2 ~ConsumerControlMarshaller	1188
6.228.3 Member Function Documentation	1188
6.228.3.1 createObject	1188
6.228.3.2 getDataStructureType	1188
6.228.3.3 looseMarshal	1188
6.228.3.4 looseUnmarshal	1189
6.228.3.5 tightMarshal1	1189
6.228.3.6 tightMarshal2	1189
6.228.3.7 tightUnmarshal	1190
6.229activemq::commands::ConsumerId Class Reference	1190
6.229.1 Member Typedef Documentation	1192
6.229.1.1 COMPARATOR	1192
6.229.2 Constructor & Destructor Documentation	1192
6.229.2.1 ConsumerId	1192
6.229.2.2 ConsumerId	1192
6.229.2.3 ConsumerId	1192
6.229.2.4 ~ConsumerId	1192
6.229.3 Member Function Documentation	1192
6.229.3.1 cloneDataStructure	1192
6.229.3.2 compareTo	1192
6.229.3.3 copyDataStructure	1192
6.229.3.4 equals	1192
6.229.3.5 equals	1193
6.229.3.6 getConnectionId	1193
6.229.3.7 getConnectionId	1193
6.229.3.8 getDataStructureType	1193
6.229.3.9 getParentId	1194
6.229.3.10getSessionId	1194
6.229.3.11getValue	1194
6.229.3.12operator<	1194

6.229.3.13	operator=	1194
6.229.3.14	operator==	1194
6.229.3.15	setConnectionId	1194
6.229.3.16	setSessionId	1194
6.229.3.17	setValue	1194
6.229.3.18	toString	1194
6.229.4	Field Documentation	1194
6.229.4.1	connectionId	1194
6.229.4.2	ID_CONSUMERID	1194
6.229.4.3	sessionId	1194
6.229.4.4	value	1194
6.230	activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller Class Reference	1195
6.230.1	Detailed Description	1196
6.230.2	Constructor & Destructor Documentation	1196
6.230.2.1	ConsumerIdMarshaller	1196
6.230.2.2	~ConsumerIdMarshaller	1196
6.230.3	Member Function Documentation	1196
6.230.3.1	createObject	1196
6.230.3.2	getDataStructureType	1196
6.230.3.3	looseMarshal	1196
6.230.3.4	looseUnmarshal	1197
6.230.3.5	tightMarshal1	1197
6.230.3.6	tightMarshal2	1198
6.230.3.7	tightUnmarshal	1198
6.231	activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller Class Reference	1198
6.231.1	Detailed Description	1199
6.231.2	Constructor & Destructor Documentation	1200
6.231.2.1	ConsumerIdMarshaller	1200
6.231.2.2	~ConsumerIdMarshaller	1200
6.231.3	Member Function Documentation	1200
6.231.3.1	createObject	1200
6.231.3.2	getDataStructureType	1200
6.231.3.3	looseMarshal	1200
6.231.3.4	looseUnmarshal	1201
6.231.3.5	tightMarshal1	1201
6.231.3.6	tightMarshal2	1201

6.231.3.7 tightUnmarshal	1202
6.232activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller Class Reference	1202
6.232.1 Detailed Description	1203
6.232.2 Constructor & Destructor Documentation	1204
6.232.2.1 ConsumerIdMarshaller	1204
6.232.2.2 ~ConsumerIdMarshaller	1204
6.232.3 Member Function Documentation	1204
6.232.3.1 createObject	1204
6.232.3.2 getDataStructureType	1204
6.232.3.3 looseMarshal	1204
6.232.3.4 looseUnmarshal	1205
6.232.3.5 tightMarshal1	1205
6.232.3.6 tightMarshal2	1205
6.232.3.7 tightUnmarshal	1206
6.233activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller Class Reference	1206
6.233.1 Detailed Description	1207
6.233.2 Constructor & Destructor Documentation	1208
6.233.2.1 ConsumerIdMarshaller	1208
6.233.2.2 ~ConsumerIdMarshaller	1208
6.233.3 Member Function Documentation	1208
6.233.3.1 createObject	1208
6.233.3.2 getDataStructureType	1208
6.233.3.3 looseMarshal	1208
6.233.3.4 looseUnmarshal	1209
6.233.3.5 tightMarshal1	1209
6.233.3.6 tightMarshal2	1209
6.233.3.7 tightUnmarshal	1210
6.234activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller Class Reference	1210
6.234.1 Detailed Description	1211
6.234.2 Constructor & Destructor Documentation	1212
6.234.2.1 ConsumerIdMarshaller	1212
6.234.2.2 ~ConsumerIdMarshaller	1212
6.234.3 Member Function Documentation	1212
6.234.3.1 createObject	1212
6.234.3.2 getDataStructureType	1212
6.234.3.3 looseMarshal	1212

6.234.3.4 looseUnmarshal	1213
6.234.3.5 tightMarshal1	1213
6.234.3.6 tightMarshal2	1213
6.234.3.7 tightUnmarshal	1214
6.235activemq::commands::ConsumerInfo Class Reference	1214
6.235.1 Constructor & Destructor Documentation	1217
6.235.1.1 ConsumerInfo	1217
6.235.1.2 ConsumerInfo	1217
6.235.1.3 ~ConsumerInfo	1217
6.235.2 Member Function Documentation	1217
6.235.2.1 cloneDataStructure	1217
6.235.2.2 copyDataStructure	1217
6.235.2.3 equals	1217
6.235.2.4 getAdditionalPredicate	1218
6.235.2.5 getAdditionalPredicate	1218
6.235.2.6 getBrokerPath	1218
6.235.2.7 getBrokerPath	1218
6.235.2.8 getConsumerId	1218
6.235.2.9 getConsumerId	1218
6.235.2.10getDataStructureType	1218
6.235.2.11getDestination	1219
6.235.2.12getDestination	1219
6.235.2.13getMaximumPendingMessageLimit	1219
6.235.2.14getNetworkConsumerPath	1219
6.235.2.15getNetworkConsumerPath	1219
6.235.2.16getPrefetchSize	1219
6.235.2.17getPriority	1219
6.235.2.18getSelector	1219
6.235.2.19getSelector	1219
6.235.2.20getSubscriptionName	1219
6.235.2.21getSubscriptionName	1219
6.235.2.22sBrowser	1219
6.235.2.23sConsumerInfo	1219
6.235.2.24sDispatchAsync	1220
6.235.2.25sExclusive	1220
6.235.2.26sNetworkSubscription	1220

6.235.2.27	isNoLocal	1220
6.235.2.28	isNoRangeAcks	1220
6.235.2.29	isOptimizedAcknowledge	1220
6.235.2.30	isRetroactive	1220
6.235.2.31	operator=	1220
6.235.2.32	setAdditionalPredicate	1220
6.235.2.33	setBrokerPath	1220
6.235.2.34	setBrowser	1220
6.235.2.35	setConsumerId	1220
6.235.2.36	setDestination	1220
6.235.2.37	setDispatchAsync	1220
6.235.2.38	setExclusive	1220
6.235.2.39	setMaximumPendingMessageLimit	1220
6.235.2.40	setNetworkConsumerPath	1220
6.235.2.41	setNetworkSubscription	1220
6.235.2.42	setNoLocal	1220
6.235.2.43	setNoRangeAcks	1220
6.235.2.44	setOptimizedAcknowledge	1220
6.235.2.45	setPrefetchSize	1220
6.235.2.46	setPriority	1220
6.235.2.47	setRetroactive	1220
6.235.2.48	setSelector	1220
6.235.2.49	setSubscriptionName	1220
6.235.2.50	toString	1220
6.235.2.51	visit	1221
6.235.3	Field Documentation	1222
6.235.3.1	additionalPredicate	1222
6.235.3.2	brokerPath	1222
6.235.3.3	browser	1222
6.235.3.4	consumerId	1222
6.235.3.5	destination	1222
6.235.3.6	dispatchAsync	1222
6.235.3.7	exclusive	1222
6.235.3.8	ID_CONSUMERINFO	1222
6.235.3.9	maximumPendingMessageLimit	1222
6.235.3.10	networkConsumerPath	1222

6.235.3.1	networkSubscription	1222
6.235.3.12	noLocal	1222
6.235.3.13	noRangeAcks	1222
6.235.3.14	optimizedAcknowledge	1222
6.235.3.15	prefetchSize	1222
6.235.3.16	priority	1222
6.235.3.17	retroactive	1222
6.235.3.18	selector	1222
6.235.3.19	subscriptionName	1222
6.236	activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller Class Reference	1223
6.236.1	Detailed Description	1224
6.236.2	Constructor & Destructor Documentation	1224
6.236.2.1	ConsumerInfoMarshaller	1224
6.236.2.2	~ConsumerInfoMarshaller	1224
6.236.3	Member Function Documentation	1224
6.236.3.1	createObject	1224
6.236.3.2	getDataStructureType	1224
6.236.3.3	looseMarshal	1224
6.236.3.4	looseUnmarshal	1225
6.236.3.5	tightMarshal1	1225
6.236.3.6	tightMarshal2	1226
6.236.3.7	tightUnmarshal	1226
6.237	activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller Class Reference	1226
6.237.1	Detailed Description	1227
6.237.2	Constructor & Destructor Documentation	1228
6.237.2.1	ConsumerInfoMarshaller	1228
6.237.2.2	~ConsumerInfoMarshaller	1228
6.237.3	Member Function Documentation	1228
6.237.3.1	createObject	1228
6.237.3.2	getDataStructureType	1228
6.237.3.3	looseMarshal	1228
6.237.3.4	looseUnmarshal	1229
6.237.3.5	tightMarshal1	1229
6.237.3.6	tightMarshal2	1229
6.237.3.7	tightUnmarshal	1230

6.238activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller Class Reference	1230
6.238.1 Detailed Description	1231
6.238.2 Constructor & Destructor Documentation	1232
6.238.2.1 ConsumerInfoMarshaller	1232
6.238.2.2 ~ConsumerInfoMarshaller	1232
6.238.3 Member Function Documentation	1232
6.238.3.1 createObject	1232
6.238.3.2 getDataStructureType	1232
6.238.3.3 looseMarshal	1232
6.238.3.4 looseUnmarshal	1233
6.238.3.5 tightMarshal1	1233
6.238.3.6 tightMarshal2	1233
6.238.3.7 tightUnmarshal	1234
6.239activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller Class Reference	1234
6.239.1 Detailed Description	1235
6.239.2 Constructor & Destructor Documentation	1236
6.239.2.1 ConsumerInfoMarshaller	1236
6.239.2.2 ~ConsumerInfoMarshaller	1236
6.239.3 Member Function Documentation	1236
6.239.3.1 createObject	1236
6.239.3.2 getDataStructureType	1236
6.239.3.3 looseMarshal	1236
6.239.3.4 looseUnmarshal	1237
6.239.3.5 tightMarshal1	1237
6.239.3.6 tightMarshal2	1237
6.239.3.7 tightUnmarshal	1238
6.240activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller Class Reference	1238
6.240.1 Detailed Description	1239
6.240.2 Constructor & Destructor Documentation	1240
6.240.2.1 ConsumerInfoMarshaller	1240
6.240.2.2 ~ConsumerInfoMarshaller	1240
6.240.3 Member Function Documentation	1240
6.240.3.1 createObject	1240
6.240.3.2 getDataStructureType	1240

6.240.3.3 looseMarshal	1240
6.240.3.4 looseUnmarshal	1241
6.240.3.5 tightMarshal1	1241
6.240.3.6 tightMarshal2	1241
6.240.3.7 tightUnmarshal	1242
6.241activemq::state::ConsumerState Class Reference	1242
6.241.1 Constructor & Destructor Documentation	1243
6.241.1.1 ConsumerState	1243
6.241.1.2 ~ConsumerState	1243
6.241.2 Member Function Documentation	1243
6.241.2.1 getInfo	1243
6.241.2.2 toString	1243
6.242activemq::commands::ControlCommand Class Reference	1243
6.242.1 Constructor & Destructor Documentation	1244
6.242.1.1 ControlCommand	1244
6.242.1.2 ControlCommand	1244
6.242.1.3 ~ControlCommand	1244
6.242.2 Member Function Documentation	1244
6.242.2.1 cloneDataStructure	1244
6.242.2.2 copyDataStructure	1244
6.242.2.3 equals	1245
6.242.2.4 getCommand	1245
6.242.2.5 getCommand	1245
6.242.2.6 getDataStructureType	1245
6.242.2.7 operator=	1245
6.242.2.8 setCommand	1245
6.242.2.9 toString	1245
6.242.2.10visit	1246
6.242.3 Field Documentation	1246
6.242.3.1 command	1246
6.242.3.2 ID_CONTROLCOMMAND	1246
6.243activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller Class Reference	1246
6.243.1 Detailed Description	1247
6.243.2 Constructor & Destructor Documentation	1247
6.243.2.1 ControlCommandMarshaller	1247
6.243.2.2 ~ControlCommandMarshaller	1247

6.243.3 Member Function Documentation	1247
6.243.3.1 createObject	1247
6.243.3.2 getDataStructureType	1248
6.243.3.3 looseMarshal	1248
6.243.3.4 looseUnmarshal	1248
6.243.3.5 tightMarshal1	1249
6.243.3.6 tightMarshal2	1249
6.243.3.7 tightUnmarshal	1250
6.244activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller Class	
Reference	1250
6.244.1 Detailed Description	1251
6.244.2 Constructor & Destructor Documentation	1251
6.244.2.1 ControlCommandMarshaller	1251
6.244.2.2 ~ControlCommandMarshaller	1251
6.244.3 Member Function Documentation	1251
6.244.3.1 createObject	1251
6.244.3.2 getDataStructureType	1252
6.244.3.3 looseMarshal	1252
6.244.3.4 looseUnmarshal	1252
6.244.3.5 tightMarshal1	1253
6.244.3.6 tightMarshal2	1253
6.244.3.7 tightUnmarshal	1253
6.245activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller Class	
Reference	1254
6.245.1 Detailed Description	1255
6.245.2 Constructor & Destructor Documentation	1255
6.245.2.1 ControlCommandMarshaller	1255
6.245.2.2 ~ControlCommandMarshaller	1255
6.245.3 Member Function Documentation	1255
6.245.3.1 createObject	1255
6.245.3.2 getDataStructureType	1255
6.245.3.3 looseMarshal	1256
6.245.3.4 looseUnmarshal	1256
6.245.3.5 tightMarshal1	1256
6.245.3.6 tightMarshal2	1257
6.245.3.7 tightUnmarshal	1257

6.246	activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller	Class	
	Reference		1258
6.246.1	Detailed Description		1259
6.246.2	Constructor & Destructor Documentation		1259
	6.246.2.1 ControlCommandMarshaller		1259
	6.246.2.2 ~ControlCommandMarshaller		1259
6.246.3	Member Function Documentation		1259
	6.246.3.1 createObject		1259
	6.246.3.2 getDataStructureType		1259
	6.246.3.3 looseMarshal		1260
	6.246.3.4 looseUnmarshal		1260
	6.246.3.5 tightMarshal1		1260
	6.246.3.6 tightMarshal2		1261
	6.246.3.7 tightUnmarshal		1261
6.247	activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller	Class	
	Reference		1262
6.247.1	Detailed Description		1263
6.247.2	Constructor & Destructor Documentation		1263
	6.247.2.1 ControlCommandMarshaller		1263
	6.247.2.2 ~ControlCommandMarshaller		1263
6.247.3	Member Function Documentation		1263
	6.247.3.1 createObject		1263
	6.247.3.2 getDataStructureType		1263
	6.247.3.3 looseMarshal		1264
	6.247.3.4 looseUnmarshal		1264
	6.247.3.5 tightMarshal1		1264
	6.247.3.6 tightMarshal2		1265
	6.247.3.7 tightUnmarshal		1265
6.248	decaf::util::concurrent::CountDownLatch	Class Reference	1266
6.248.1	Constructor & Destructor Documentation		1266
	6.248.1.1 CountDownLatch		1266
	6.248.1.2 ~CountDownLatch		1267
6.248.2	Member Function Documentation		1267
	6.248.2.1 await		1267
	6.248.2.2 await		1267
	6.248.2.3 countDown		1267
	6.248.2.4 getCount		1267

6.249	activemq::commands::DataArrayResponse Class Reference	1267
6.249.1	Constructor & Destructor Documentation	1269
6.249.1.1	DataArrayResponse	1269
6.249.1.2	DataArrayResponse	1269
6.249.1.3	~DataArrayResponse	1269
6.249.2	Member Function Documentation	1269
6.249.2.1	cloneDataStructure	1269
6.249.2.2	copyDataStructure	1269
6.249.2.3	equals	1269
6.249.2.4	getData	1270
6.249.2.5	getData	1270
6.249.2.6	getDataStructureType	1270
6.249.2.7	operator=	1270
6.249.2.8	setData	1270
6.249.2.9	toString	1270
6.249.3	Field Documentation	1270
6.249.3.1	data	1270
6.249.3.2	ID_DATAARRAYRESPONSE	1270
6.250	activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller Class Reference	1271
6.250.1	Detailed Description	1272
6.250.2	Constructor & Destructor Documentation	1272
6.250.2.1	DataArrayResponseMarshaller	1272
6.250.2.2	~DataArrayResponseMarshaller	1272
6.250.3	Member Function Documentation	1272
6.250.3.1	createObject	1272
6.250.3.2	getDataStructureType	1272
6.250.3.3	looseMarshal	1272
6.250.3.4	looseUnmarshal	1273
6.250.3.5	tightMarshal1	1273
6.250.3.6	tightMarshal2	1274
6.250.3.7	tightUnmarshal	1274
6.251	activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller Class Reference	1275
6.251.1	Detailed Description	1276
6.251.2	Constructor & Destructor Documentation	1276
6.251.2.1	DataArrayResponseMarshaller	1276

6.251.2.2	~DataArrayResponseMarshaller	1276
6.251.3	Member Function Documentation	1276
6.251.3.1	createObject	1276
6.251.3.2	getDataStructureType	1276
6.251.3.3	looseMarshal	1276
6.251.3.4	looseUnmarshal	1277
6.251.3.5	tightMarshal1	1277
6.251.3.6	tightMarshal2	1278
6.251.3.7	tightUnmarshal	1278
6.252	activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller Class	
	Reference	1279
6.252.1	Detailed Description	1280
6.252.2	Constructor & Destructor Documentation	1280
6.252.2.1	DataArrayResponseMarshaller	1280
6.252.2.2	~DataArrayResponseMarshaller	1280
6.252.3	Member Function Documentation	1280
6.252.3.1	createObject	1280
6.252.3.2	getDataStructureType	1280
6.252.3.3	looseMarshal	1280
6.252.3.4	looseUnmarshal	1281
6.252.3.5	tightMarshal1	1281
6.252.3.6	tightMarshal2	1282
6.252.3.7	tightUnmarshal	1282
6.253	activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller Class	
	Reference	1283
6.253.1	Detailed Description	1284
6.253.2	Constructor & Destructor Documentation	1284
6.253.2.1	DataArrayResponseMarshaller	1284
6.253.2.2	~DataArrayResponseMarshaller	1284
6.253.3	Member Function Documentation	1284
6.253.3.1	createObject	1284
6.253.3.2	getDataStructureType	1284
6.253.3.3	looseMarshal	1284
6.253.3.4	looseUnmarshal	1285
6.253.3.5	tightMarshal1	1285
6.253.3.6	tightMarshal2	1286
6.253.3.7	tightUnmarshal	1286

6.254	activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller Class	
	Reference	1287
6.254.1	Detailed Description	1288
6.254.2	Constructor & Destructor Documentation	1288
	6.254.2.1 DataArrayResponseMarshaller	1288
	6.254.2.2 ~DataArrayResponseMarshaller	1288
6.254.3	Member Function Documentation	1288
	6.254.3.1 createObject	1288
	6.254.3.2 getDataStructureType	1288
	6.254.3.3 looseMarshal	1288
	6.254.3.4 looseUnmarshal	1289
	6.254.3.5 tightMarshal1	1289
	6.254.3.6 tightMarshal2	1290
	6.254.3.7 tightUnmarshal	1290
6.255	decaf::io::DataInputStream Class Reference	1291
6.255.1	Detailed Description	1292
6.255.2	Constructor & Destructor Documentation	1293
	6.255.2.1 DataInputStream	1293
	6.255.2.2 ~DataInputStream	1293
6.255.3	Member Function Documentation	1293
	6.255.3.1 read	1293
	6.255.3.2 read	1293
	6.255.3.3 read	1294
	6.255.3.4 readBoolean	1295
	6.255.3.5 readByte	1295
	6.255.3.6 readChar	1295
	6.255.3.7 readDouble	1296
	6.255.3.8 readFloat	1296
	6.255.3.9 readFully	1296
	6.255.3.10 readFully	1297
	6.255.3.11 readInt	1297
	6.255.3.12 readLong	1298
	6.255.3.13 readShort	1298
	6.255.3.14 readString	1298
	6.255.3.15 readUnsignedByte	1299
	6.255.3.16 readUnsignedShort	1299
	6.255.3.17 readUTF	1299

6.255.3.18skip	1300
6.256decaf::io::DataOutputStream Class Reference	1300
6.256.1 Detailed Description	1302
6.256.2 Constructor & Destructor Documentation	1302
6.256.2.1 DataOutputStream	1302
6.256.2.2 ~DataOutputStream	1302
6.256.3 Member Function Documentation	1302
6.256.3.1 size	1302
6.256.3.2 write	1302
6.256.3.3 write	1303
6.256.3.4 write	1303
6.256.3.5 writeBoolean	1303
6.256.3.6 writeByte	1304
6.256.3.7 writeBytes	1304
6.256.3.8 writeChar	1304
6.256.3.9 writeChars	1304
6.256.3.10writeDouble	1305
6.256.3.11writeFloat	1305
6.256.3.12writeInt	1305
6.256.3.13writeLong	1306
6.256.3.14writeShort	1306
6.256.3.15writeUnsignedShort	1306
6.256.3.16writeUTF	1306
6.256.4 Field Documentation	1307
6.256.4.1 buffer	1307
6.256.4.2 written	1307
6.257activemq::commands::DataResponse Class Reference	1307
6.257.1 Constructor & Destructor Documentation	1308
6.257.1.1 DataResponse	1308
6.257.1.2 DataResponse	1308
6.257.1.3 ~DataResponse	1308
6.257.2 Member Function Documentation	1308
6.257.2.1 cloneDataStructure	1308
6.257.2.2 copyDataStructure	1308
6.257.2.3 equals	1309
6.257.2.4 getData	1309

6.257.2.5	getData	1309
6.257.2.6	getDataStructureType	1309
6.257.2.7	operator=	1309
6.257.2.8	setData	1309
6.257.2.9	toString	1309
6.257.3	Field Documentation	1310
6.257.3.1	data	1310
6.257.3.2	ID_DATARESPONSE	1310
6.258	activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller Class Reference	1310
6.258.1	Detailed Description	1311
6.258.2	Constructor & Destructor Documentation	1311
6.258.2.1	DataResponseMarshaller	1311
6.258.2.2	~DataResponseMarshaller	1311
6.258.3	Member Function Documentation	1311
6.258.3.1	createObject	1311
6.258.3.2	getDataStructureType	1311
6.258.3.3	looseMarshal	1312
6.258.3.4	looseUnmarshal	1312
6.258.3.5	tightMarshal1	1312
6.258.3.6	tightMarshal2	1313
6.258.3.7	tightUnmarshal	1313
6.259	activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller Class Reference	1314
6.259.1	Detailed Description	1315
6.259.2	Constructor & Destructor Documentation	1315
6.259.2.1	DataResponseMarshaller	1315
6.259.2.2	~DataResponseMarshaller	1315
6.259.3	Member Function Documentation	1315
6.259.3.1	createObject	1315
6.259.3.2	getDataStructureType	1315
6.259.3.3	looseMarshal	1316
6.259.3.4	looseUnmarshal	1316
6.259.3.5	tightMarshal1	1316
6.259.3.6	tightMarshal2	1317
6.259.3.7	tightUnmarshal	1317

6.260	activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller Class Reference	1318
6.260.1	Detailed Description	1319
6.260.2	Constructor & Destructor Documentation	1319
6.260.2.1	DataResponseMarshaller	1319
6.260.2.2	~DataResponseMarshaller	1319
6.260.3	Member Function Documentation	1319
6.260.3.1	createObject	1319
6.260.3.2	getDataStructureType	1319
6.260.3.3	looseMarshal	1320
6.260.3.4	looseUnmarshal	1320
6.260.3.5	tightMarshal1	1320
6.260.3.6	tightMarshal2	1321
6.260.3.7	tightUnmarshal	1321
6.261	activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller Class Reference	1322
6.261.1	Detailed Description	1323
6.261.2	Constructor & Destructor Documentation	1323
6.261.2.1	DataResponseMarshaller	1323
6.261.2.2	~DataResponseMarshaller	1323
6.261.3	Member Function Documentation	1323
6.261.3.1	createObject	1323
6.261.3.2	getDataStructureType	1323
6.261.3.3	looseMarshal	1324
6.261.3.4	looseUnmarshal	1324
6.261.3.5	tightMarshal1	1324
6.261.3.6	tightMarshal2	1325
6.261.3.7	tightUnmarshal	1325
6.262	activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller Class Reference	1326
6.262.1	Detailed Description	1327
6.262.2	Constructor & Destructor Documentation	1327
6.262.2.1	DataResponseMarshaller	1327
6.262.2.2	~DataResponseMarshaller	1327
6.262.3	Member Function Documentation	1327
6.262.3.1	createObject	1327
6.262.3.2	getDataStructureType	1327

6.262.3.3 looseMarshal	1328
6.262.3.4 looseUnmarshal	1328
6.262.3.5 tightMarshal1	1328
6.262.3.6 tightMarshal2	1329
6.262.3.7 tightUnmarshal	1329
6.263activemq::wireformat::openwire::marshal::DataStreamMarshaller Class Reference	1330
6.263.1 Detailed Description	1331
6.263.2 Constructor & Destructor Documentation	1331
6.263.2.1 ~DataStreamMarshaller	1331
6.263.3 Member Function Documentation	1331
6.263.3.1 createObject	1331
6.263.3.2 getDataStructureType	1336
6.263.3.3 looseMarshal	1342
6.263.3.4 looseUnmarshal	1348
6.263.3.5 tightMarshal1	1354
6.263.3.6 tightMarshal2	1360
6.263.3.7 tightUnmarshal	1366
6.264activemq::commands::DataStructure Class Reference	1372
6.264.1 Constructor & Destructor Documentation	1373
6.264.1.1 ~DataStructure	1373
6.264.2 Member Function Documentation	1373
6.264.2.1 cloneDataStructure	1373
6.264.2.2 copyDataStructure	1374
6.264.2.3 equals	1374
6.264.2.4 getDataStructureType	1375
6.264.2.5 toString	1376
6.265decaf::util::Date Class Reference	1377
6.265.1 Detailed Description	1378
6.265.2 Constructor & Destructor Documentation	1378
6.265.2.1 Date	1378
6.265.2.2 Date	1378
6.265.2.3 Date	1379
6.265.2.4 ~Date	1379
6.265.3 Member Function Documentation	1379
6.265.3.1 after	1379
6.265.3.2 before	1379

6.265.3.3	compareTo	1379
6.265.3.4	equals	1380
6.265.3.5	getTime	1380
6.265.3.6	operator<	1380
6.265.3.7	operator=	1380
6.265.3.8	operator==	1380
6.265.3.9	setTime	1381
6.265.3.10	toString	1381
6.266	decaf::internal::DecafRuntime Class Reference	1381
6.266.1	Detailed Description	1382
6.266.2	Constructor & Destructor Documentation	1382
6.266.2.1	DecafRuntime	1382
6.266.2.2	~DecafRuntime	1382
6.266.3	Member Function Documentation	1382
6.266.3.1	getGlobalPool	1382
6.267	activemq::threads::DedicatedTaskRunner Class Reference	1382
6.267.1	Constructor & Destructor Documentation	1383
6.267.1.1	DedicatedTaskRunner	1383
6.267.1.2	~DedicatedTaskRunner	1383
6.267.2	Member Function Documentation	1383
6.267.2.1	run	1383
6.267.2.2	shutdown	1383
6.267.2.3	shutdown	1383
6.267.2.4	wakeup	1384
6.268	activemq::transport::DefaultTransportListener Class Reference	1384
6.268.1	Constructor & Destructor Documentation	1384
6.268.1.1	~DefaultTransportListener	1384
6.268.2	Member Function Documentation	1384
6.268.2.1	onCommand	1384
6.268.2.2	onException	1385
6.268.2.3	transportInterrupted	1385
6.268.2.4	transportResumed	1385
6.269	decaf::util::concurrent::Delayed Class Reference	1385
6.269.1	Detailed Description	1386
6.269.2	Constructor & Destructor Documentation	1386
6.269.2.1	~Delayed	1386

6.269.3 Member Function Documentation	1386
6.269.3.1 getDelay	1386
6.270cms::DeliveryMode Class Reference	1386
6.270.1 Detailed Description	1387
6.270.2 Member Enumeration Documentation	1387
6.270.2.1 DELIVERY_MODE	1387
6.270.3 Constructor & Destructor Documentation	1387
6.270.3.1 ~DeliveryMode	1387
6.271cms::Destination Class Reference	1387
6.271.1 Detailed Description	1388
6.271.2 Member Enumeration Documentation	1388
6.271.2.1 DestinationType	1388
6.271.3 Constructor & Destructor Documentation	1389
6.271.3.1 ~Destination	1389
6.271.4 Member Function Documentation	1389
6.271.4.1 clone	1389
6.271.4.2 copy	1389
6.271.4.3 getCMSProperties	1389
6.271.4.4 getDestinationType	1389
6.272activemq::commands::ActiveMQDestination::DestinationFilter Struct Reference . .	1390
6.272.1 Field Documentation	1390
6.272.1.1 ANY_CHILD	1390
6.272.1.2 ANY_DESCENDENT	1390
6.273activemq::commands::DestinationInfo Class Reference	1390
6.273.1 Constructor & Destructor Documentation	1392
6.273.1.1 DestinationInfo	1392
6.273.1.2 DestinationInfo	1392
6.273.1.3 ~DestinationInfo	1392
6.273.2 Member Function Documentation	1392
6.273.2.1 cloneDataStructure	1392
6.273.2.2 copyDataStructure	1392
6.273.2.3 equals	1392
6.273.2.4 getBrokerPath	1393
6.273.2.5 getBrokerPath	1393
6.273.2.6 getConnectionId	1393
6.273.2.7 getConnectionId	1393

6.273.2.8	getDataStructureType	1393
6.273.2.9	getDestination	1394
6.273.2.10	getDestination	1394
6.273.2.11	getOperationType	1394
6.273.2.12	getTimeout	1394
6.273.2.13	operator=	1394
6.273.2.14	setBrokerPath	1394
6.273.2.15	setConnectionId	1394
6.273.2.16	setDestination	1394
6.273.2.17	setOperationType	1394
6.273.2.18	setTimeout	1394
6.273.2.19	toString	1394
6.273.2.20	visit	1394
6.273.3	Field Documentation	1395
6.273.3.1	brokerPath	1395
6.273.3.2	connectionId	1395
6.273.3.3	destination	1395
6.273.3.4	ID_DESTINATIONINFO	1395
6.273.3.5	operationType	1395
6.273.3.6	timeout	1395
6.274	activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller Class Reference	1395
6.274.1	Detailed Description	1396
6.274.2	Constructor & Destructor Documentation	1396
6.274.2.1	DestinationInfoMarshaller	1396
6.274.2.2	~DestinationInfoMarshaller	1396
6.274.3	Member Function Documentation	1396
6.274.3.1	createObject	1396
6.274.3.2	getDataStructureType	1397
6.274.3.3	looseMarshal	1397
6.274.3.4	looseUnmarshal	1397
6.274.3.5	tightMarshal1	1398
6.274.3.6	tightMarshal2	1398
6.274.3.7	tightUnmarshal	1399
6.275	activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller Class Reference	1399
6.275.1	Detailed Description	1400

6.275.2 Constructor & Destructor Documentation	1400
6.275.2.1 DestinationInfoMarshaller	1400
6.275.2.2 ~DestinationInfoMarshaller	1400
6.275.3 Member Function Documentation	1400
6.275.3.1 createObject	1400
6.275.3.2 getDataStructureType	1401
6.275.3.3 looseMarshal	1401
6.275.3.4 looseUnmarshal	1401
6.275.3.5 tightMarshal1	1402
6.275.3.6 tightMarshal2	1402
6.275.3.7 tightUnmarshal	1402
6.276activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller Class Reference	1403
6.276.1 Detailed Description	1404
6.276.2 Constructor & Destructor Documentation	1404
6.276.2.1 DestinationInfoMarshaller	1404
6.276.2.2 ~DestinationInfoMarshaller	1404
6.276.3 Member Function Documentation	1404
6.276.3.1 createObject	1404
6.276.3.2 getDataStructureType	1404
6.276.3.3 looseMarshal	1405
6.276.3.4 looseUnmarshal	1405
6.276.3.5 tightMarshal1	1405
6.276.3.6 tightMarshal2	1406
6.276.3.7 tightUnmarshal	1406
6.277activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller Class Reference	1407
6.277.1 Detailed Description	1408
6.277.2 Constructor & Destructor Documentation	1408
6.277.2.1 DestinationInfoMarshaller	1408
6.277.2.2 ~DestinationInfoMarshaller	1408
6.277.3 Member Function Documentation	1408
6.277.3.1 createObject	1408
6.277.3.2 getDataStructureType	1408
6.277.3.3 looseMarshal	1409
6.277.3.4 looseUnmarshal	1409
6.277.3.5 tightMarshal1	1409

6.277.3.6 tightMarshal2	1410
6.277.3.7 tightUnmarshal	1410
6.278activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller Class Reference	1411
6.278.1 Detailed Description	1412
6.278.2 Constructor & Destructor Documentation	1412
6.278.2.1 DestinationInfoMarshaller	1412
6.278.2.2 ~DestinationInfoMarshaller	1412
6.278.3 Member Function Documentation	1412
6.278.3.1 createObject	1412
6.278.3.2 getDataStructureType	1412
6.278.3.3 looseMarshal	1413
6.278.3.4 looseUnmarshal	1413
6.278.3.5 tightMarshal1	1413
6.278.3.6 tightMarshal2	1414
6.278.3.7 tightUnmarshal	1414
6.279activemq::cmsutil::DestinationResolver Class Reference	1415
6.279.1 Detailed Description	1415
6.279.2 Constructor & Destructor Documentation	1415
6.279.2.1 ~DestinationResolver	1415
6.279.3 Member Function Documentation	1415
6.279.3.1 destroy	1415
6.279.3.2 init	1416
6.279.3.3 resolveDestinationName	1416
6.280activemq::commands::DiscoveryEvent Class Reference	1416
6.280.1 Constructor & Destructor Documentation	1418
6.280.1.1 DiscoveryEvent	1418
6.280.1.2 DiscoveryEvent	1418
6.280.1.3 ~DiscoveryEvent	1418
6.280.2 Member Function Documentation	1418
6.280.2.1 cloneDataStructure	1418
6.280.2.2 copyDataStructure	1418
6.280.2.3 equals	1418
6.280.2.4 getBrokerName	1419
6.280.2.5 getBrokerName	1419
6.280.2.6 getDataStructureType	1419
6.280.2.7 getServiceName	1419

6.280.2.8 getServiceName	1419
6.280.2.9 operator=	1419
6.280.2.10 setBrokerName	1419
6.280.2.11 setServiceName	1419
6.280.2.12 toString	1419
6.280.3 Field Documentation	1420
6.280.3.1 brokerName	1420
6.280.3.2 ID_DISCOVERYEVENT	1420
6.280.3.3 serviceName	1420
6.281activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller Class Reference	1420
6.281.1 Detailed Description	1421
6.281.2 Constructor & Destructor Documentation	1421
6.281.2.1 DiscoveryEventMarshaller	1421
6.281.2.2 ~DiscoveryEventMarshaller	1421
6.281.3 Member Function Documentation	1421
6.281.3.1 createObject	1421
6.281.3.2 getDataStructureType	1421
6.281.3.3 looseMarshal	1422
6.281.3.4 looseUnmarshal	1422
6.281.3.5 tightMarshal1	1422
6.281.3.6 tightMarshal2	1423
6.281.3.7 tightUnmarshal	1423
6.282activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller Class Reference	1424
6.282.1 Detailed Description	1425
6.282.2 Constructor & Destructor Documentation	1425
6.282.2.1 DiscoveryEventMarshaller	1425
6.282.2.2 ~DiscoveryEventMarshaller	1425
6.282.3 Member Function Documentation	1425
6.282.3.1 createObject	1425
6.282.3.2 getDataStructureType	1425
6.282.3.3 looseMarshal	1425
6.282.3.4 looseUnmarshal	1426
6.282.3.5 tightMarshal1	1426
6.282.3.6 tightMarshal2	1427
6.282.3.7 tightUnmarshal	1427

6.283activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller Class Reference	1427
6.283.1 Detailed Description	1428
6.283.2 Constructor & Destructor Documentation	1429
6.283.2.1 DiscoveryEventMarshaller	1429
6.283.2.2 ~DiscoveryEventMarshaller	1429
6.283.3 Member Function Documentation	1429
6.283.3.1 createObject	1429
6.283.3.2 getDataStructureType	1429
6.283.3.3 looseMarshal	1429
6.283.3.4 looseUnmarshal	1430
6.283.3.5 tightMarshal1	1430
6.283.3.6 tightMarshal2	1430
6.283.3.7 tightUnmarshal	1431
6.284activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller Class Reference	1431
6.284.1 Detailed Description	1432
6.284.2 Constructor & Destructor Documentation	1433
6.284.2.1 DiscoveryEventMarshaller	1433
6.284.2.2 ~DiscoveryEventMarshaller	1433
6.284.3 Member Function Documentation	1433
6.284.3.1 createObject	1433
6.284.3.2 getDataStructureType	1433
6.284.3.3 looseMarshal	1433
6.284.3.4 looseUnmarshal	1434
6.284.3.5 tightMarshal1	1434
6.284.3.6 tightMarshal2	1434
6.284.3.7 tightUnmarshal	1435
6.285activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller Class Reference	1435
6.285.1 Detailed Description	1436
6.285.2 Constructor & Destructor Documentation	1437
6.285.2.1 DiscoveryEventMarshaller	1437
6.285.2.2 ~DiscoveryEventMarshaller	1437
6.285.3 Member Function Documentation	1437
6.285.3.1 createObject	1437
6.285.3.2 getDataStructureType	1437

6.285.3.3 looseMarshal	1437
6.285.3.4 looseUnmarshal	1438
6.285.3.5 tightMarshal1	1438
6.285.3.6 tightMarshal2	1438
6.285.3.7 tightUnmarshal	1439
6.286activemq::core::DispatchData Class Reference	1439
6.286.1 Detailed Description	1440
6.286.2 Constructor & Destructor Documentation	1440
6.286.2.1 DispatchData	1440
6.286.2.2 DispatchData	1440
6.286.3 Member Function Documentation	1440
6.286.3.1 getConsumerId	1440
6.286.3.2 getMessage	1440
6.287activemq::core::Dispatcher Class Reference	1440
6.287.1 Detailed Description	1440
6.287.2 Constructor & Destructor Documentation	1441
6.287.2.1 ~Dispatcher	1441
6.287.3 Member Function Documentation	1441
6.287.3.1 dispatch	1441
6.288decaf::lang::Double Class Reference	1441
6.288.1 Constructor & Destructor Documentation	1443
6.288.1.1 Double	1443
6.288.1.2 Double	1443
6.288.1.3 ~Double	1444
6.288.2 Member Function Documentation	1444
6.288.2.1 byteValue	1444
6.288.2.2 compare	1444
6.288.2.3 compareTo	1444
6.288.2.4 compareTo	1444
6.288.2.5 doubleToLongBits	1445
6.288.2.6 doubleToRawLongBits	1445
6.288.2.7 doubleValue	1446
6.288.2.8 equals	1446
6.288.2.9 equals	1446
6.288.2.1float Value	1446
6.288.2.1 lint Value	1447

6.288.2.12sInfinite	1447
6.288.2.13sInfinite	1447
6.288.2.14sNaN	1447
6.288.2.15sNaN	1447
6.288.2.16ongBitsToDouble	1447
6.288.2.17ongValue	1448
6.288.2.18operator<	1448
6.288.2.19operator<	1448
6.288.2.20operator==	1449
6.288.2.21operator==	1449
6.288.2.22parseDouble	1449
6.288.2.23hortValue	1449
6.288.2.24oHexString	1450
6.288.2.25oString	1450
6.288.2.26oString	1450
6.288.2.27valueOf	1451
6.288.2.28valueOf	1451
6.288.3 Field Documentation	1451
6.288.3.1 MAX_VALUE	1451
6.288.3.2 MIN_VALUE	1451
6.288.3.3 NaN	1452
6.288.3.4 NEGATIVE_INFINITY	1452
6.288.3.5 POSITIVE_INFINITY	1452
6.288.3.6 SIZE	1452
6.289decaf::internal::nio::DoubleArrayBuffer Class Reference	1452
6.289.1 Constructor & Destructor Documentation	1454
6.289.1.1 DoubleArrayBuffer	1454
6.289.1.2 DoubleArrayBuffer	1454
6.289.1.3 DoubleArrayBuffer	1454
6.289.1.4 DoubleArrayBuffer	1455
6.289.1.5 ~DoubleArrayBuffer	1455
6.289.2 Member Function Documentation	1455
6.289.2.1 array	1455
6.289.2.2 arrayOffset	1455
6.289.2.3 asReadOnlyBuffer	1456
6.289.2.4 compact	1456

6.289.2.5 duplicate	1456
6.289.2.6 get	1457
6.289.2.7 get	1457
6.289.2.8 hasArray	1457
6.289.2.9 isReadOnly	1458
6.289.2.10put	1458
6.289.2.11put	1458
6.289.2.12setReadOnly	1459
6.289.2.13slice	1459
6.290decaf::nio::DoubleBuffer Class Reference	1459
6.290.1 Detailed Description	1461
6.290.2 Constructor & Destructor Documentation	1462
6.290.2.1 DoubleBuffer	1462
6.290.2.2 ~DoubleBuffer	1462
6.290.3 Member Function Documentation	1462
6.290.3.1 allocate	1462
6.290.3.2 array	1462
6.290.3.3 arrayOffset	1463
6.290.3.4 asReadOnlyBuffer	1463
6.290.3.5 compact	1464
6.290.3.6 compareTo	1464
6.290.3.7 duplicate	1464
6.290.3.8 equals	1465
6.290.3.9 get	1465
6.290.3.10get	1465
6.290.3.11get	1465
6.290.3.12get	1466
6.290.3.13hasArray	1466
6.290.3.14operator<	1467
6.290.3.15operator==	1467
6.290.3.16put	1467
6.290.3.17put	1468
6.290.3.18put	1468
6.290.3.19put	1469
6.290.3.20put	1469
6.290.3.21slice	1469

6.290.3.22	toString	1470
6.290.3.23	wrap	1470
6.290.3.24	wrap	1470
6.291	decaf::lang::DYNAMIC_CAST_TOKEN Struct Reference	1471
6.292	activemq::cmsutil::DynamicDestinationResolver Class Reference	1471
6.292.1	Detailed Description	1472
6.292.2	Constructor & Destructor Documentation	1472
6.292.2.1	~DynamicDestinationResolver	1472
6.292.3	Member Function Documentation	1472
6.292.3.1	destroy	1472
6.292.3.2	init	1472
6.292.3.3	resolveDestinationName	1472
6.293	decaf::util::Map< K, V, COMPARATOR >::Entry Class Reference	1473
6.293.1	Constructor & Destructor Documentation	1473
6.293.1.1	Entry	1473
6.293.1.2	~Entry	1473
6.293.2	Member Function Documentation	1473
6.293.2.1	getKey	1473
6.293.2.2	getValue	1473
6.293.2.3	setValue	1473
6.294	decaf::io::EOFException Class Reference	1474
6.294.1	Constructor & Destructor Documentation	1474
6.294.1.1	EOFException	1474
6.294.1.2	EOFException	1474
6.294.1.3	EOFException	1475
6.294.1.4	EOFException	1475
6.294.1.5	EOFException	1475
6.294.1.6	EOFException	1475
6.294.1.7	~EOFException	1476
6.294.2	Member Function Documentation	1476
6.294.2.1	clone	1476
6.295	decaf::lang::Exception Class Reference	1476
6.295.1	Constructor & Destructor Documentation	1478
6.295.1.1	Exception	1478
6.295.1.2	Exception	1478
6.295.1.3	Exception	1478

6.295.1.4 Exception	1478
6.295.1.5 Exception	1479
6.295.1.6 ~Exception	1479
6.295.2 Member Function Documentation	1479
6.295.2.1 buildMessage	1479
6.295.2.2 clone	1479
6.295.2.3 getCause	1480
6.295.2.4 getMessage	1480
6.295.2.5 getStackTrace	1480
6.295.2.6 getStackTraceString	1481
6.295.2.7 initCause	1481
6.295.2.8 operator=	1481
6.295.2.9 printStackTrace	1481
6.295.2.10 printStackTrace	1481
6.295.2.11 setMark	1482
6.295.2.12 setMessage	1482
6.295.2.13 setStackTrace	1482
6.295.2.14 what	1482
6.295.3 Field Documentation	1482
6.295.3.1 cause	1482
6.295.3.2 message	1482
6.295.3.3 stackTrace	1483
6.296 cms::ExceptionListener Class Reference	1483
6.296.1 Detailed Description	1483
6.296.2 Constructor & Destructor Documentation	1483
6.296.2.1 ~ExceptionListener	1483
6.296.3 Member Function Documentation	1483
6.296.3.1 onException	1483
6.297 activemq::commands::ExceptionResponse Class Reference	1484
6.297.1 Constructor & Destructor Documentation	1485
6.297.1.1 ExceptionResponse	1485
6.297.1.2 ExceptionResponse	1485
6.297.1.3 ~ExceptionResponse	1485
6.297.2 Member Function Documentation	1485
6.297.2.1 cloneDataStructure	1485
6.297.2.2 copyDataStructure	1485

6.297.2.3 equals	1485
6.297.2.4 getDataStructureType	1486
6.297.2.5 getException	1486
6.297.2.6 getException	1486
6.297.2.7 operator=	1486
6.297.2.8 setException	1486
6.297.2.9 toString	1486
6.297.3 Field Documentation	1486
6.297.3.1 exception	1486
6.297.3.2 ID_EXCEPTIONRESPONSE	1486
6.298activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller Class	
Reference	1487
6.298.1 Detailed Description	1488
6.298.2 Constructor & Destructor Documentation	1488
6.298.2.1 ExceptionResponseMarshaller	1488
6.298.2.2 ~ExceptionResponseMarshaller	1488
6.298.3 Member Function Documentation	1488
6.298.3.1 createObject	1488
6.298.3.2 getDataStructureType	1488
6.298.3.3 looseMarshal	1488
6.298.3.4 looseUnmarshal	1489
6.298.3.5 tightMarshal1	1489
6.298.3.6 tightMarshal2	1490
6.298.3.7 tightUnmarshal	1490
6.299activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller Class	
Reference	1491
6.299.1 Detailed Description	1492
6.299.2 Constructor & Destructor Documentation	1492
6.299.2.1 ExceptionResponseMarshaller	1492
6.299.2.2 ~ExceptionResponseMarshaller	1492
6.299.3 Member Function Documentation	1492
6.299.3.1 createObject	1492
6.299.3.2 getDataStructureType	1492
6.299.3.3 looseMarshal	1492
6.299.3.4 looseUnmarshal	1493
6.299.3.5 tightMarshal1	1493
6.299.3.6 tightMarshal2	1494

6.299.3.7 tightUnmarshal	1494
6.300activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller Class	
Reference	1495
6.300.1 Detailed Description	1496
6.300.2 Constructor & Destructor Documentation	1496
6.300.2.1 ExceptionResponseMarshaller	1496
6.300.2.2 ~ExceptionResponseMarshaller	1496
6.300.3 Member Function Documentation	1496
6.300.3.1 createObject	1496
6.300.3.2 getDataStructureType	1496
6.300.3.3 looseMarshal	1496
6.300.3.4 looseUnmarshal	1497
6.300.3.5 tightMarshal1	1497
6.300.3.6 tightMarshal2	1498
6.300.3.7 tightUnmarshal	1498
6.301activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller Class	
Reference	1499
6.301.1 Detailed Description	1500
6.301.2 Constructor & Destructor Documentation	1500
6.301.2.1 ExceptionResponseMarshaller	1500
6.301.2.2 ~ExceptionResponseMarshaller	1500
6.301.3 Member Function Documentation	1500
6.301.3.1 createObject	1500
6.301.3.2 getDataStructureType	1500
6.301.3.3 looseMarshal	1500
6.301.3.4 looseUnmarshal	1501
6.301.3.5 tightMarshal1	1501
6.301.3.6 tightMarshal2	1502
6.301.3.7 tightUnmarshal	1502
6.302activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller Class	
Reference	1503
6.302.1 Detailed Description	1504
6.302.2 Constructor & Destructor Documentation	1504
6.302.2.1 ExceptionResponseMarshaller	1504
6.302.2.2 ~ExceptionResponseMarshaller	1504
6.302.3 Member Function Documentation	1504
6.302.3.1 createObject	1504

6.302.3.2	getDataStructureType	1504
6.302.3.3	looseMarshal	1504
6.302.3.4	looseUnmarshal	1505
6.302.3.5	tightMarshal1	1505
6.302.3.6	tightMarshal2	1506
6.302.3.7	tightUnmarshal	1506
6.303	decaf::util::concurrent::ExecutionException Class Reference	1507
6.303.1	Constructor & Destructor Documentation	1507
6.303.1.1	ExecutionException	1507
6.303.1.2	ExecutionException	1507
6.303.1.3	ExecutionException	1508
6.303.1.4	ExecutionException	1508
6.303.1.5	ExecutionException	1508
6.303.1.6	ExecutionException	1508
6.303.1.7	~ExecutionException	1509
6.303.2	Member Function Documentation	1509
6.303.2.1	clone	1509
6.304	decaf::util::concurrent::Executor Class Reference	1509
6.304.1	Detailed Description	1509
6.304.2	Constructor & Destructor Documentation	1510
6.304.2.1	~Executor	1510
6.304.3	Member Function Documentation	1510
6.304.3.1	execute	1510
6.305	decaf::util::concurrent::ExecutorService Class Reference	1511
6.305.1	Detailed Description	1511
6.305.2	Constructor & Destructor Documentation	1512
6.305.2.1	~ExecutorService	1512
6.305.3	Member Function Documentation	1512
6.305.3.1	awaitTermination	1512
6.306	activemq::transport::failover::FailoverTransport Class Reference	1512
6.306.1	Constructor & Destructor Documentation	1515
6.306.1.1	FailoverTransport	1515
6.306.1.2	~FailoverTransport	1515
6.306.2	Member Function Documentation	1515
6.306.2.1	add	1515
6.306.2.2	addURI	1515

6.306.2.3 close	1515
6.306.2.4 getBackOffMultiplier	1516
6.306.2.5 getBackupPoolSize	1516
6.306.2.6 getInitialReconnectDelay	1516
6.306.2.7 getMaxCacheSize	1516
6.306.2.8 getMaxReconnectAttempts	1516
6.306.2.9 getMaxReconnectDelay	1516
6.306.2.10getReconnectDelay	1516
6.306.2.11getRemoteAddress	1516
6.306.2.12getTimeout	1516
6.306.2.13getTransportListener	1516
6.306.2.14handleTransportFailure	1517
6.306.2.15sBackup	1517
6.306.2.16sClosed	1517
6.306.2.17sConnected	1517
6.306.2.18sFault Tolerant	1517
6.306.2.19sInitialized	1518
6.306.2.20sPending	1518
6.306.2.21sRandomize	1518
6.306.2.22sTrackMessages	1518
6.306.2.23sUseExponentialBackOff	1518
6.306.2.24terate	1518
6.306.2.25narrow	1518
6.306.2.26neway	1519
6.306.2.27reconnect	1519
6.306.2.28reconnect	1519
6.306.2.29removeURI	1519
6.306.2.30request	1520
6.306.2.31request	1520
6.306.2.32restoreTransport	1520
6.306.2.33etBackOffMultiplier	1521
6.306.2.34etBackup	1521
6.306.2.35etBackupPoolSize	1521
6.306.2.36etInitialized	1521
6.306.2.37etInitialReconnectDelay	1521
6.306.2.38etMaxCacheSize	1521

6.306.2.39	setMaxReconnectAttempts	1521
6.306.2.40	setMaxReconnectDelay	1521
6.306.2.41	setRandomize	1521
6.306.2.42	setReconnectDelay	1521
6.306.2.43	setTimeout	1521
6.306.2.44	setTrackMessages	1521
6.306.2.45	setTransportListener	1521
6.306.2.46	setUseExponentialBackOff	1522
6.306.2.47	setWireFormat	1522
6.306.2.48	start	1522
6.306.2.49	stop	1522
6.306.3	Friends And Related Function Documentation	1522
6.306.3.1	FailoverTransportListener	1522
6.307	activemq::transport::failover::FailoverTransportFactory Class Reference	1523
6.307.1	Detailed Description	1523
6.307.2	Constructor & Destructor Documentation	1524
6.307.2.1	~FailoverTransportFactory	1524
6.307.3	Member Function Documentation	1524
6.307.3.1	create	1524
6.307.3.2	createComposite	1524
6.307.3.3	doCreateComposite	1524
6.308	activemq::transport::failover::FailoverTransportListener Class Reference	1525
6.308.1	Detailed Description	1525
6.308.2	Constructor & Destructor Documentation	1526
6.308.2.1	FailoverTransportListener	1526
6.308.2.2	~FailoverTransportListener	1526
6.308.3	Member Function Documentation	1526
6.308.3.1	onCommand	1526
6.308.3.2	onException	1526
6.308.3.3	transportInterrupted	1526
6.308.3.4	transportResumed	1526
6.309	decaf::util::logging::Filter Class Reference	1527
6.309.1	Detailed Description	1527
6.309.2	Constructor & Destructor Documentation	1527
6.309.2.1	~Filter	1527
6.309.3	Member Function Documentation	1527

6.309.3.1 isLoggable	1527
6.310decaf::io::FilterInputStream Class Reference	1528
6.310.1 Detailed Description	1529
6.310.2 Constructor & Destructor Documentation	1530
6.310.2.1 FilterInputStream	1530
6.310.2.2 ~FilterInputStream	1530
6.310.3 Member Function Documentation	1530
6.310.3.1 available	1530
6.310.3.2 close	1530
6.310.3.3 isClosed	1531
6.310.3.4 lock	1531
6.310.3.5 mark	1531
6.310.3.6 markSupported	1531
6.310.3.7 notify	1532
6.310.3.8 notifyAll	1532
6.310.3.9 read	1532
6.310.3.10 read	1533
6.310.3.11 reset	1533
6.310.3.12 skip	1534
6.310.3.13 tryLock	1534
6.310.3.14 unlock	1535
6.310.3.15 wait	1535
6.310.3.16 wait	1535
6.310.3.17 wait	1536
6.310.4 Field Documentation	1536
6.310.4.1 closed	1536
6.310.4.2 inputStream	1536
6.310.4.3 mutex	1536
6.310.4.4 own	1536
6.311decaf::io::FilterOutputStream Class Reference	1536
6.311.1 Detailed Description	1538
6.311.2 Constructor & Destructor Documentation	1538
6.311.2.1 FilterOutputStream	1538
6.311.2.2 ~FilterOutputStream	1538
6.311.3 Member Function Documentation	1539
6.311.3.1 close	1539

6.311.3.2 flush	1539
6.311.3.3 isClosed	1539
6.311.3.4 lock	1539
6.311.3.5 notify	1540
6.311.3.6 notifyAll	1540
6.311.3.7 tryLock	1540
6.311.3.8 unlock	1540
6.311.3.9 wait	1541
6.311.3.10wait	1541
6.311.3.11wait	1541
6.311.3.12write	1542
6.311.3.13write	1542
6.311.3.14write	1543
6.311.4 Field Documentation	1543
6.311.4.1 closed	1543
6.311.4.2 mutex	1543
6.311.4.3 outputStream	1543
6.311.4.4 own	1543
6.312decaf::lang::Float Class Reference	1544
6.312.1 Constructor & Destructor Documentation	1546
6.312.1.1 Float	1546
6.312.1.2 Float	1546
6.312.1.3 Float	1546
6.312.1.4 ~Float	1546
6.312.2 Member Function Documentation	1546
6.312.2.1 byteValue	1546
6.312.2.2 compare	1547
6.312.2.3 compareTo	1547
6.312.2.4 compareTo	1547
6.312.2.5 doubleValue	1548
6.312.2.6 equals	1548
6.312.2.7 equals	1548
6.312.2.8 floatToIntBits	1548
6.312.2.9 floatToRawIntBits	1549
6.312.2.10float Value	1549
6.312.2.11intBitsToFloat	1549

6.312.2.12	nt Value	1550
6.312.2.13	sInfinite	1550
6.312.2.14	sInfinite	1550
6.312.2.15	sNaN	1550
6.312.2.16	sNaN	1550
6.312.2.17	ongValue	1550
6.312.2.18	operator<	1551
6.312.2.19	operator<	1551
6.312.2.20	operator==	1551
6.312.2.21	operator==	1551
6.312.2.22	parseFloat	1552
6.312.2.23	hortValue	1552
6.312.2.24	oHexString	1552
6.312.2.25	oString	1553
6.312.2.26	oString	1553
6.312.2.27	valueOf	1553
6.312.2.28	valueOf	1554
6.312.3	Field Documentation	1554
6.312.3.1	MAX_VALUE	1554
6.312.3.2	MIN_VALUE	1554
6.312.3.3	NaN	1554
6.312.3.4	NEGATIVE_INFINITY	1554
6.312.3.5	POSITIVE_INFINITY	1554
6.312.3.6	SIZE	1554
6.313	decaf::internal::nio::FloatArrayBuffer Class Reference	1555
6.313.1	Constructor & Destructor Documentation	1556
6.313.1.1	FloatArrayBuffer	1556
6.313.1.2	FloatArrayBuffer	1556
6.313.1.3	FloatArrayBuffer	1557
6.313.1.4	FloatArrayBuffer	1557
6.313.1.5	~FloatArrayBuffer	1557
6.313.2	Member Function Documentation	1557
6.313.2.1	array	1557
6.313.2.2	arrayOffset	1558
6.313.2.3	asReadOnlyBuffer	1558
6.313.2.4	compact	1559

6.313.2.5 duplicate	1559
6.313.2.6 get	1559
6.313.2.7 get	1560
6.313.2.8 hasArray	1560
6.313.2.9 isReadOnly	1560
6.313.2.10put	1560
6.313.2.11put	1561
6.313.2.12setReadOnly	1561
6.313.2.13slice	1561
6.314decaf::nio::FloatBuffer Class Reference	1562
6.314.1 Detailed Description	1564
6.314.2 Constructor & Destructor Documentation	1564
6.314.2.1 FloatBuffer	1564
6.314.2.2 ~FloatBuffer	1564
6.314.3 Member Function Documentation	1564
6.314.3.1 allocate	1564
6.314.3.2 array	1565
6.314.3.3 arrayOffset	1565
6.314.3.4 asReadOnlyBuffer	1565
6.314.3.5 compact	1566
6.314.3.6 compareTo	1566
6.314.3.7 duplicate	1567
6.314.3.8 equals	1567
6.314.3.9 get	1567
6.314.3.10get	1567
6.314.3.11get	1568
6.314.3.12get	1568
6.314.3.13hasArray	1569
6.314.3.14operator<	1569
6.314.3.15operator==	1569
6.314.3.16put	1569
6.314.3.17put	1570
6.314.3.18put	1570
6.314.3.19put	1571
6.314.3.20put	1571
6.314.3.21slice	1572

6.314.3.22 toString	1572
6.314.3.23 wrap	1572
6.314.3.24 wrap	1572
6.315activemq::commands::FlushCommand Class Reference	1573
6.315.1 Constructor & Destructor Documentation	1574
6.315.1.1 FlushCommand	1574
6.315.1.2 FlushCommand	1574
6.315.1.3 ~FlushCommand	1574
6.315.2 Member Function Documentation	1574
6.315.2.1 cloneDataStructure	1574
6.315.2.2 copyDataStructure	1574
6.315.2.3 equals	1574
6.315.2.4 getDataStructureType	1575
6.315.2.5 operator=	1575
6.315.2.6 toString	1575
6.315.2.7 visit	1575
6.315.3 Field Documentation	1575
6.315.3.1 ID_FLUSHCOMMAND	1575
6.316activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller Class Reference	1576
6.316.1 Detailed Description	1577
6.316.2 Constructor & Destructor Documentation	1577
6.316.2.1 FlushCommandMarshaller	1577
6.316.2.2 ~FlushCommandMarshaller	1577
6.316.3 Member Function Documentation	1577
6.316.3.1 createObject	1577
6.316.3.2 getDataStructureType	1577
6.316.3.3 looseMarshal	1577
6.316.3.4 looseUnmarshal	1578
6.316.3.5 tightMarshal1	1578
6.316.3.6 tightMarshal2	1579
6.316.3.7 tightUnmarshal	1579
6.317activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller Class Reference	1579
6.317.1 Detailed Description	1580
6.317.2 Constructor & Destructor Documentation	1581
6.317.2.1 FlushCommandMarshaller	1581

6.317.2.2	~FlushCommandMarshaller	1581
6.317.3	Member Function Documentation	1581
6.317.3.1	createObject	1581
6.317.3.2	getDataStructureType	1581
6.317.3.3	looseMarshal	1581
6.317.3.4	looseUnmarshal	1582
6.317.3.5	tightMarshal1	1582
6.317.3.6	tightMarshal2	1582
6.317.3.7	tightUnmarshal	1583
6.318	activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller Class Reference	1583
6.318.1	Detailed Description	1584
6.318.2	Constructor & Destructor Documentation	1585
6.318.2.1	FlushCommandMarshaller	1585
6.318.2.2	~FlushCommandMarshaller	1585
6.318.3	Member Function Documentation	1585
6.318.3.1	createObject	1585
6.318.3.2	getDataStructureType	1585
6.318.3.3	looseMarshal	1585
6.318.3.4	looseUnmarshal	1586
6.318.3.5	tightMarshal1	1586
6.318.3.6	tightMarshal2	1586
6.318.3.7	tightUnmarshal	1587
6.319	activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller Class Reference	1587
6.319.1	Detailed Description	1588
6.319.2	Constructor & Destructor Documentation	1589
6.319.2.1	FlushCommandMarshaller	1589
6.319.2.2	~FlushCommandMarshaller	1589
6.319.3	Member Function Documentation	1589
6.319.3.1	createObject	1589
6.319.3.2	getDataStructureType	1589
6.319.3.3	looseMarshal	1589
6.319.3.4	looseUnmarshal	1590
6.319.3.5	tightMarshal1	1590
6.319.3.6	tightMarshal2	1590
6.319.3.7	tightUnmarshal	1591

6.320activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller Class Reference	1591
6.320.1 Detailed Description	1592
6.320.2 Constructor & Destructor Documentation	1593
6.320.2.1 FlushCommandMarshaller	1593
6.320.2.2 ~FlushCommandMarshaller	1593
6.320.3 Member Function Documentation	1593
6.320.3.1 createObject	1593
6.320.3.2 getDataStructureType	1593
6.320.3.3 looseMarshal	1593
6.320.3.4 looseUnmarshal	1594
6.320.3.5 tightMarshal1	1594
6.320.3.6 tightMarshal2	1594
6.320.3.7 tightUnmarshal	1595
6.321decaf::util::logging::Formatter Class Reference	1595
6.321.1 Detailed Description	1596
6.321.2 Constructor & Destructor Documentation	1596
6.321.2.1 ~Formatter	1596
6.321.3 Member Function Documentation	1596
6.321.3.1 format	1596
6.321.3.2 formatMessage	1596
6.321.3.3 getHead	1597
6.321.3.4 getTail	1597
6.322decaf::util::concurrent::Future< V > Class Template Reference	1597
6.322.1 Detailed Description	1598
6.322.2 Constructor & Destructor Documentation	1598
6.322.2.1 ~Future	1598
6.322.3 Member Function Documentation	1598
6.322.3.1 cancel	1598
6.322.3.2 get	1599
6.322.3.3 get	1599
6.322.3.4 isCancelled	1600
6.322.3.5 isDone	1600
6.323activemq::transport::correlator::FutureResponse Class Reference	1600
6.323.1 Detailed Description	1600
6.323.2 Constructor & Destructor Documentation	1601
6.323.2.1 FutureResponse	1601

6.323.2.2 ~FutureResponse	1601
6.323.3 Member Function Documentation	1601
6.323.3.1 getResponse	1601
6.323.3.2 getResponse	1601
6.323.3.3 getResponse	1601
6.323.3.4 getResponse	1601
6.323.3.5 setResponse	1601
6.324decaf::security::GeneralSecurityException Class Reference	1602
6.324.1 Constructor & Destructor Documentation	1602
6.324.1.1 GeneralSecurityException	1602
6.324.1.2 GeneralSecurityException	1603
6.324.1.3 GeneralSecurityException	1603
6.324.1.4 GeneralSecurityException	1603
6.324.1.5 GeneralSecurityException	1603
6.324.1.6 GeneralSecurityException	1603
6.324.1.7 ~GeneralSecurityException	1604
6.324.2 Member Function Documentation	1604
6.324.2.1 clone	1604
6.325decaf::util::logging::Handler Class Reference	1604
6.325.1 Detailed Description	1605
6.325.2 Constructor & Destructor Documentation	1605
6.325.2.1 ~Handler	1605
6.325.3 Member Function Documentation	1605
6.325.3.1 flush	1605
6.325.3.2 getFilter	1606
6.325.3.3 getFormatter	1606
6.325.3.4 getLevel	1606
6.325.3.5 isLoggable	1606
6.325.3.6 publish	1606
6.325.3.7 setFilter	1607
6.325.3.8 setFormatter	1607
6.325.3.9 setLevel	1607
6.326decaf::internal::util::HexStringParser Class Reference	1607
6.326.1 Constructor & Destructor Documentation	1608
6.326.1.1 HexStringParser	1608
6.326.1.2 ~HexStringParser	1608

6.326.2 Member Function Documentation	1608
6.326.2.1 parse	1608
6.326.2.2 parseDouble	1609
6.326.2.3 parseFloat	1609
6.327activemq::wireformat::openwire::utils::HexTable Class Reference	1609
6.327.1 Detailed Description	1609
6.327.2 Constructor & Destructor Documentation	1610
6.327.2.1 HexTable	1610
6.327.2.2 ~HexTable	1610
6.327.3 Member Function Documentation	1610
6.327.3.1 operator[]	1610
6.327.3.2 operator[]	1610
6.327.3.3 size	1610
6.328decaf::net::HttpRetryException Class Reference	1610
6.328.1 Constructor & Destructor Documentation	1611
6.328.1.1 HttpRetryException	1611
6.328.1.2 HttpRetryException	1611
6.328.1.3 HttpRetryException	1611
6.328.1.4 HttpRetryException	1612
6.328.1.5 HttpRetryException	1612
6.328.1.6 HttpRetryException	1612
6.328.1.7 ~HttpRetryException	1613
6.328.2 Member Function Documentation	1613
6.328.2.1 clone	1613
6.329decaf::lang::exceptions::IllegalArgumentException Class Reference	1613
6.329.1 Constructor & Destructor Documentation	1614
6.329.1.1 IllegalArgumentException	1614
6.329.1.2 IllegalArgumentException	1614
6.329.1.3 IllegalArgumentException	1614
6.329.1.4 IllegalArgumentException	1614
6.329.1.5 IllegalArgumentException	1614
6.329.1.6 IllegalArgumentException	1615
6.329.1.7 ~IllegalArgumentException	1615
6.329.2 Member Function Documentation	1615
6.329.2.1 clone	1615
6.330decaf::lang::exceptions::IllegalMonitorStateException Class Reference	1616

6.330.1 Constructor & Destructor Documentation	1616
6.330.1.1 IllegalMonitorStateException	1616
6.330.1.2 IllegalMonitorStateException	1616
6.330.1.3 IllegalMonitorStateException	1617
6.330.1.4 IllegalMonitorStateException	1617
6.330.1.5 IllegalMonitorStateException	1617
6.330.1.6 IllegalMonitorStateException	1617
6.330.1.7 ~IllegalMonitorStateException	1618
6.330.2 Member Function Documentation	1618
6.330.2.1 clone	1618
6.331decaf::lang::exceptions::IllegalStateException Class Reference	1618
6.331.1 Constructor & Destructor Documentation	1619
6.331.1.1 IllegalStateException	1619
6.331.1.2 IllegalStateException	1619
6.331.1.3 IllegalStateException	1619
6.331.1.4 IllegalStateException	1619
6.331.1.5 IllegalStateException	1620
6.331.1.6 IllegalStateException	1620
6.331.1.7 ~IllegalStateException	1620
6.331.2 Member Function Documentation	1620
6.331.2.1 clone	1620
6.332cms::IllegalStateException Class Reference	1621
6.332.1 Detailed Description	1621
6.332.2 Constructor & Destructor Documentation	1621
6.332.2.1 IllegalStateException	1621
6.332.2.2 IllegalStateException	1621
6.332.2.3 IllegalStateException	1621
6.332.2.4 IllegalStateException	1621
6.332.2.5 ~IllegalStateException	1621
6.333decaf::lang::exceptions::IllegalThreadStateException Class Reference	1622
6.333.1 Constructor & Destructor Documentation	1622
6.333.1.1 IllegalThreadStateException	1622
6.333.1.2 IllegalThreadStateException	1622
6.333.1.3 IllegalThreadStateException	1623
6.333.1.4 IllegalThreadStateException	1623
6.333.1.5 IllegalThreadStateException	1623

6.333.1.6	IllegalThreadStateException	1623
6.333.1.7	~IllegalThreadStateException	1624
6.333.2	Member Function Documentation	1624
6.333.2.1	clone	1624
6.334	activemq::transport::inactivity::InactivityMonitor Class Reference	1624
6.334.1	Constructor & Destructor Documentation	1625
6.334.1.1	InactivityMonitor	1625
6.334.1.2	InactivityMonitor	1625
6.334.1.3	~InactivityMonitor	1625
6.334.2	Member Function Documentation	1625
6.334.2.1	close	1625
6.334.2.2	getInitialDelayTime	1626
6.334.2.3	getReadCheckTime	1626
6.334.2.4	getWriteCheckTime	1626
6.334.2.5	isKeepAliveResponseRequired	1626
6.334.2.6	onCommand	1626
6.334.2.7	oneway	1626
6.334.2.8	onException	1626
6.334.2.9	setInitialDelayTime	1627
6.334.2.10	setKeepAliveResponseRequired	1627
6.334.2.11	setReadCheckTime	1627
6.334.2.12	setWriteCheckTime	1627
6.334.3	Friends And Related Function Documentation	1627
6.334.3.1	AsyncSignalReadErrorTask	1627
6.334.3.2	AsyncWriteTask	1627
6.334.3.3	ReadChecker	1627
6.334.3.4	WriteChecker	1627
6.335	decaf::lang::exceptions::IndexOutOfBoundsException Class Reference	1627
6.335.1	Constructor & Destructor Documentation	1628
6.335.1.1	IndexOutOfBoundsException	1628
6.335.1.2	IndexOutOfBoundsException	1628
6.335.1.3	IndexOutOfBoundsException	1628
6.335.1.4	IndexOutOfBoundsException	1629
6.335.1.5	IndexOutOfBoundsException	1629
6.335.1.6	IndexOutOfBoundsException	1629
6.335.1.7	~IndexOutOfBoundsException	1629

6.335.2 Member Function Documentation	1629
6.335.2.1 clone	1629
6.336decaf::io::InputStream Class Reference	1630
6.336.1 Detailed Description	1631
6.336.2 Constructor & Destructor Documentation	1631
6.336.2.1 ~InputStream	1631
6.336.3 Member Function Documentation	1631
6.336.3.1 available	1631
6.336.3.2 mark	1631
6.336.3.3 markSupported	1631
6.336.3.4 read	1632
6.336.3.5 read	1632
6.336.3.6 reset	1633
6.336.3.7 skip	1633
6.337decaf::internal::nio::IntArrayBuffer Class Reference	1634
6.337.1 Constructor & Destructor Documentation	1635
6.337.1.1 IntArrayBuffer	1635
6.337.1.2 IntArrayBuffer	1636
6.337.1.3 IntArrayBuffer	1636
6.337.1.4 IntArrayBuffer	1636
6.337.1.5 ~IntArrayBuffer	1637
6.337.2 Member Function Documentation	1637
6.337.2.1 array	1637
6.337.2.2 arrayOffset	1637
6.337.2.3 asReadOnlyBuffer	1638
6.337.2.4 compact	1638
6.337.2.5 duplicate	1638
6.337.2.6 get	1639
6.337.2.7 get	1639
6.337.2.8 hasArray	1639
6.337.2.9 isReadOnly	1640
6.337.2.10put	1640
6.337.2.11put	1640
6.337.2.12setReadOnly	1641
6.337.2.13slice	1641
6.338decaf::nio::IntBuffer Class Reference	1641

6.338.1 Detailed Description	1643
6.338.2 Constructor & Destructor Documentation	1644
6.338.2.1 IntBuffer	1644
6.338.2.2 ~IntBuffer	1644
6.338.3 Member Function Documentation	1644
6.338.3.1 allocate	1644
6.338.3.2 array	1644
6.338.3.3 arrayOffset	1645
6.338.3.4 asReadOnlyBuffer	1645
6.338.3.5 compact	1645
6.338.3.6 compareTo	1646
6.338.3.7 duplicate	1646
6.338.3.8 equals	1646
6.338.3.9 get	1646
6.338.3.10 get	1647
6.338.3.11 get	1647
6.338.3.12 get	1648
6.338.3.13 hasArray	1648
6.338.3.14 operator<	1648
6.338.3.15 operator==	1649
6.338.3.16 put	1649
6.338.3.17 put	1649
6.338.3.18 put	1650
6.338.3.19 put	1650
6.338.3.20 put	1650
6.338.3.21 slice	1651
6.338.3.22 toString	1651
6.338.3.23 wrap	1651
6.338.3.24 wrap	1652
6.339 decaf::lang::Integer Class Reference	1652
6.339.1 Constructor & Destructor Documentation	1655
6.339.1.1 Integer	1655
6.339.1.2 Integer	1655
6.339.1.3 ~Integer	1656
6.339.2 Member Function Documentation	1656
6.339.2.1 bitCount	1656

6.339.2.2	byteValue	1656
6.339.2.3	compareTo	1656
6.339.2.4	compareTo	1656
6.339.2.5	decode	1657
6.339.2.6	doubleValue	1657
6.339.2.7	equals	1657
6.339.2.8	equals	1658
6.339.2.9	float Value	1658
6.339.2.10	highestOneBit	1658
6.339.2.11	int Value	1658
6.339.2.12	long Value	1658
6.339.2.13	lowestOneBit	1659
6.339.2.14	numberOfLeadingZeros	1659
6.339.2.15	numberOfTrailingZeros	1659
6.339.2.16	operator<	1660
6.339.2.17	operator<	1660
6.339.2.18	operator==	1660
6.339.2.19	operator==	1660
6.339.2.20	parseInt	1661
6.339.2.21	parseInt	1661
6.339.2.22	reverse	1662
6.339.2.23	reverseBytes	1662
6.339.2.24	rotateLeft	1662
6.339.2.25	rotateRight	1662
6.339.2.26	short Value	1663
6.339.2.27	signum	1663
6.339.2.28	toBinaryString	1663
6.339.2.29	toHexString	1664
6.339.2.30	toOctalString	1664
6.339.2.31	toString	1664
6.339.2.32	toString	1665
6.339.2.33	toString	1665
6.339.2.34	valueOf	1665
6.339.2.35	valueOf	1666
6.339.2.36	valueOf	1666
6.339.3	Field Documentation	1666

6.339.3.1 MAX_VALUE	1666
6.339.3.2 MIN_VALUE	1666
6.339.3.3 SIZE	1667
6.340activemq::commands::IntegerResponse Class Reference	1667
6.340.1 Constructor & Destructor Documentation	1668
6.340.1.1 IntegerResponse	1668
6.340.1.2 IntegerResponse	1668
6.340.1.3 ~IntegerResponse	1668
6.340.2 Member Function Documentation	1668
6.340.2.1 cloneDataStructure	1668
6.340.2.2 copyDataStructure	1668
6.340.2.3 equals	1668
6.340.2.4 getDataStructureType	1669
6.340.2.5 getResult	1669
6.340.2.6 operator=	1669
6.340.2.7 setResult	1669
6.340.2.8 toString	1669
6.340.3 Field Documentation	1669
6.340.3.1 ID_INTEGERRESPONSE	1669
6.340.3.2 result	1669
6.341activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller Class Reference	1669
6.341.1 Detailed Description	1670
6.341.2 Constructor & Destructor Documentation	1671
6.341.2.1 IntegerResponseMarshaller	1671
6.341.2.2 ~IntegerResponseMarshaller	1671
6.341.3 Member Function Documentation	1671
6.341.3.1 createObject	1671
6.341.3.2 getDataStructureType	1671
6.341.3.3 looseMarshal	1671
6.341.3.4 looseUnmarshal	1672
6.341.3.5 tightMarshal1	1672
6.341.3.6 tightMarshal2	1673
6.341.3.7 tightUnmarshal	1673
6.342activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller Class Reference	1673
6.342.1 Detailed Description	1674

6.342.2 Constructor & Destructor Documentation	1675
6.342.2.1 IntegerResponseMarshaller	1675
6.342.2.2 ~IntegerResponseMarshaller	1675
6.342.3 Member Function Documentation	1675
6.342.3.1 createObject	1675
6.342.3.2 getDataStructureType	1675
6.342.3.3 looseMarshal	1675
6.342.3.4 looseUnmarshal	1676
6.342.3.5 tightMarshal1	1676
6.342.3.6 tightMarshal2	1677
6.342.3.7 tightUnmarshal	1677
6.343activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller Class	
Reference	1677
6.343.1 Detailed Description	1678
6.343.2 Constructor & Destructor Documentation	1679
6.343.2.1 IntegerResponseMarshaller	1679
6.343.2.2 ~IntegerResponseMarshaller	1679
6.343.3 Member Function Documentation	1679
6.343.3.1 createObject	1679
6.343.3.2 getDataStructureType	1679
6.343.3.3 looseMarshal	1679
6.343.3.4 looseUnmarshal	1680
6.343.3.5 tightMarshal1	1680
6.343.3.6 tightMarshal2	1681
6.343.3.7 tightUnmarshal	1681
6.344activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller Class	
Reference	1681
6.344.1 Detailed Description	1682
6.344.2 Constructor & Destructor Documentation	1683
6.344.2.1 IntegerResponseMarshaller	1683
6.344.2.2 ~IntegerResponseMarshaller	1683
6.344.3 Member Function Documentation	1683
6.344.3.1 createObject	1683
6.344.3.2 getDataStructureType	1683
6.344.3.3 looseMarshal	1683
6.344.3.4 looseUnmarshal	1684
6.344.3.5 tightMarshal1	1684

6.344.3.6	tightMarshal2	1685
6.344.3.7	tightUnmarshal	1685
6.345	activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller Class Reference	1685
6.345.1	Detailed Description	1686
6.345.2	Constructor & Destructor Documentation	1687
6.345.2.1	IntegerResponseMarshaller	1687
6.345.2.2	~IntegerResponseMarshaller	1687
6.345.3	Member Function Documentation	1687
6.345.3.1	createObject	1687
6.345.3.2	getDataStructureType	1687
6.345.3.3	looseMarshal	1687
6.345.3.4	looseUnmarshal	1688
6.345.3.5	tightMarshal1	1688
6.345.3.6	tightMarshal2	1689
6.345.3.7	tightUnmarshal	1689
6.346	activemq::transport::mock::InternalCommandListener Class Reference	1689
6.346.1	Detailed Description	1690
6.346.2	Constructor & Destructor Documentation	1690
6.346.2.1	InternalCommandListener	1690
6.346.2.2	~InternalCommandListener	1690
6.346.3	Member Function Documentation	1690
6.346.3.1	onCommand	1690
6.346.3.2	run	1691
6.346.3.3	setResponseBuilder	1691
6.346.3.4	setTransport	1691
6.347	decaf::lang::exceptions::InterruptedException Class Reference	1691
6.347.1	Constructor & Destructor Documentation	1692
6.347.1.1	InterruptedException	1692
6.347.1.2	InterruptedException	1692
6.347.1.3	InterruptedException	1692
6.347.1.4	InterruptedException	1692
6.347.1.5	InterruptedException	1692
6.347.1.6	InterruptedException	1693
6.347.1.7	~InterruptedException	1693
6.347.2	Member Function Documentation	1693
6.347.2.1	clone	1693

6.348	decaf::io::InterruptedIOException Class Reference	1693
6.348.1	Constructor & Destructor Documentation	1694
6.348.1.1	InterruptedIOException	1694
6.348.1.2	InterruptedIOException	1694
6.348.1.3	InterruptedIOException	1694
6.348.1.4	InterruptedIOException	1695
6.348.1.5	InterruptedIOException	1695
6.348.1.6	InterruptedIOException	1695
6.348.1.7	~InterruptedIOException	1695
6.348.2	Member Function Documentation	1695
6.348.2.1	clone	1695
6.349	cms::InvalidClientIdException Class Reference	1696
6.349.1	Detailed Description	1696
6.349.2	Constructor & Destructor Documentation	1697
6.349.2.1	InvalidClientIdException	1697
6.349.2.2	InvalidClientIdException	1697
6.349.2.3	InvalidClientIdException	1697
6.349.2.4	InvalidClientIdException	1697
6.349.2.5	~InvalidClientIdException	1697
6.350	cms::InvalidDestinationException Class Reference	1697
6.350.1	Detailed Description	1697
6.350.2	Constructor & Destructor Documentation	1698
6.350.2.1	InvalidDestinationException	1698
6.350.2.2	InvalidDestinationException	1698
6.350.2.3	InvalidDestinationException	1698
6.350.2.4	InvalidDestinationException	1698
6.350.2.5	~InvalidDestinationException	1698
6.351	decaf::security::InvalidKeyException Class Reference	1698
6.351.1	Constructor & Destructor Documentation	1699
6.351.1.1	InvalidKeyException	1699
6.351.1.2	InvalidKeyException	1699
6.351.1.3	InvalidKeyException	1699
6.351.1.4	InvalidKeyException	1699
6.351.1.5	InvalidKeyException	1700
6.351.1.6	InvalidKeyException	1700
6.351.1.7	~InvalidKeyException	1700

6.351.2 Member Function Documentation	1700
6.351.2.1 clone	1700
6.352decaf::nio::InvalidMarkException Class Reference	1700
6.352.1 Constructor & Destructor Documentation	1701
6.352.1.1 InvalidMarkException	1701
6.352.1.2 InvalidMarkException	1701
6.352.1.3 InvalidMarkException	1702
6.352.1.4 InvalidMarkException	1702
6.352.1.5 InvalidMarkException	1702
6.352.1.6 InvalidMarkException	1702
6.352.1.7 ~InvalidMarkException	1703
6.352.2 Member Function Documentation	1703
6.352.2.1 clone	1703
6.353cms::InvalidSelectorException Class Reference	1703
6.353.1 Detailed Description	1703
6.353.2 Constructor & Destructor Documentation	1704
6.353.2.1 InvalidSelectorException	1704
6.353.2.2 InvalidSelectorException	1704
6.353.2.3 InvalidSelectorException	1704
6.353.2.4 InvalidSelectorException	1704
6.353.2.5 ~InvalidSelectorException	1704
6.354decaf::lang::exceptions::InvalidStateException Class Reference	1704
6.354.1 Constructor & Destructor Documentation	1705
6.354.1.1 InvalidStateException	1705
6.354.1.2 InvalidStateException	1705
6.354.1.3 InvalidStateException	1705
6.354.1.4 InvalidStateException	1705
6.354.1.5 InvalidStateException	1706
6.354.1.6 InvalidStateException	1706
6.354.1.7 ~InvalidStateException	1706
6.354.2 Member Function Documentation	1706
6.354.2.1 clone	1706
6.355decaf::io::IOException Class Reference	1707
6.355.1 Constructor & Destructor Documentation	1707
6.355.1.1 IOException	1707
6.355.1.2 IOException	1707

6.355.1.3	IOException	1708
6.355.1.4	IOException	1708
6.355.1.5	IOException	1708
6.355.1.6	IOException	1708
6.355.1.7	~IOException	1709
6.355.2	Member Function Documentation	1709
6.355.2.1	clone	1709
6.356	activemq::transport::IOTransport Class Reference	1709
6.356.1	Detailed Description	1711
6.356.2	Constructor & Destructor Documentation	1711
6.356.2.1	IOTransport	1711
6.356.2.2	IOTransport	1711
6.356.2.3	~IOTransport	1711
6.356.3	Member Function Documentation	1711
6.356.3.1	close	1711
6.356.3.2	getRemoteAddress	1712
6.356.3.3	getTransportListener	1712
6.356.3.4	isClosed	1712
6.356.3.5	isConnected	1712
6.356.3.6	isFaultTolerant	1712
6.356.3.7	narrow	1712
6.356.3.8	oneway	1713
6.356.3.9	reconnect	1713
6.356.3.10	request	1713
6.356.3.11	request	1714
6.356.3.12	run	1714
6.356.3.13	setInputStream	1714
6.356.3.14	setOutputStream	1714
6.356.3.15	setTransportListener	1714
6.356.3.16	setWireFormat	1715
6.356.3.17	start	1715
6.356.3.18	stop	1715
6.357	decaf::lang::Iterable< E > Class Template Reference	1715
6.357.1	Detailed Description	1716
6.357.2	Constructor & Destructor Documentation	1716
6.357.2.1	~Iterable	1716

6.357.3 Member Function Documentation	1716
6.357.3.1 iterator	1716
6.357.3.2 iterator	1716
6.358 decaf::util::Iterator< T > Class Template Reference	1716
6.358.1 Detailed Description	1717
6.358.2 Constructor & Destructor Documentation	1717
6.358.2.1 ~Iterator	1717
6.358.3 Member Function Documentation	1717
6.358.3.1 hasNext	1717
6.358.3.2 next	1717
6.358.3.3 remove	1718
6.359 activemq::commands::JournalQueueAck Class Reference	1718
6.359.1 Constructor & Destructor Documentation	1719
6.359.1.1 JournalQueueAck	1719
6.359.1.2 JournalQueueAck	1719
6.359.1.3 ~JournalQueueAck	1719
6.359.2 Member Function Documentation	1719
6.359.2.1 cloneDataStructure	1719
6.359.2.2 copyDataStructure	1720
6.359.2.3 equals	1720
6.359.2.4 getDataStructureType	1720
6.359.2.5 getDestination	1721
6.359.2.6 getDestination	1721
6.359.2.7 getMessageAck	1721
6.359.2.8 getMessageAck	1721
6.359.2.9 operator=	1721
6.359.2.10 setDestination	1721
6.359.2.11 setMessageAck	1721
6.359.2.12 toString	1721
6.359.3 Field Documentation	1721
6.359.3.1 destination	1721
6.359.3.2 ID_JOURNALQUEUEACK	1721
6.359.3.3 messageAck	1721
6.360 activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller Class Reference	1722
6.360.1 Detailed Description	1723
6.360.2 Constructor & Destructor Documentation	1723

6.360.2.1 JournalQueueAckMarshaller	1723
6.360.2.2 ~JournalQueueAckMarshaller	1723
6.360.3 Member Function Documentation	1723
6.360.3.1 createObject	1723
6.360.3.2 getDataStructureType	1723
6.360.3.3 looseMarshal	1723
6.360.3.4 looseUnmarshal	1724
6.360.3.5 tightMarshal1	1724
6.360.3.6 tightMarshal2	1725
6.360.3.7 tightUnmarshal	1725
6.361activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller Class	
Reference	1725
6.361.1 Detailed Description	1726
6.361.2 Constructor & Destructor Documentation	1727
6.361.2.1 JournalQueueAckMarshaller	1727
6.361.2.2 ~JournalQueueAckMarshaller	1727
6.361.3 Member Function Documentation	1727
6.361.3.1 createObject	1727
6.361.3.2 getDataStructureType	1727
6.361.3.3 looseMarshal	1727
6.361.3.4 looseUnmarshal	1728
6.361.3.5 tightMarshal1	1728
6.361.3.6 tightMarshal2	1728
6.361.3.7 tightUnmarshal	1729
6.362activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller Class	
Reference	1729
6.362.1 Detailed Description	1730
6.362.2 Constructor & Destructor Documentation	1731
6.362.2.1 JournalQueueAckMarshaller	1731
6.362.2.2 ~JournalQueueAckMarshaller	1731
6.362.3 Member Function Documentation	1731
6.362.3.1 createObject	1731
6.362.3.2 getDataStructureType	1731
6.362.3.3 looseMarshal	1731
6.362.3.4 looseUnmarshal	1732
6.362.3.5 tightMarshal1	1732
6.362.3.6 tightMarshal2	1732

6.362.3.7 tightUnmarshal	1733
6.363activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller Class	
Reference	1733
6.363.1 Detailed Description	1734
6.363.2 Constructor & Destructor Documentation	1735
6.363.2.1 JournalQueueAckMarshaller	1735
6.363.2.2 ~JournalQueueAckMarshaller	1735
6.363.3 Member Function Documentation	1735
6.363.3.1 createObject	1735
6.363.3.2 getDataStructureType	1735
6.363.3.3 looseMarshal	1735
6.363.3.4 looseUnmarshal	1736
6.363.3.5 tightMarshal1	1736
6.363.3.6 tightMarshal2	1736
6.363.3.7 tightUnmarshal	1737
6.364activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller Class	
Reference	1737
6.364.1 Detailed Description	1738
6.364.2 Constructor & Destructor Documentation	1739
6.364.2.1 JournalQueueAckMarshaller	1739
6.364.2.2 ~JournalQueueAckMarshaller	1739
6.364.3 Member Function Documentation	1739
6.364.3.1 createObject	1739
6.364.3.2 getDataStructureType	1739
6.364.3.3 looseMarshal	1739
6.364.3.4 looseUnmarshal	1740
6.364.3.5 tightMarshal1	1740
6.364.3.6 tightMarshal2	1740
6.364.3.7 tightUnmarshal	1741
6.365activemq::commands::JournalTopicAck Class Reference	1741
6.365.1 Constructor & Destructor Documentation	1743
6.365.1.1 JournalTopicAck	1743
6.365.1.2 JournalTopicAck	1743
6.365.1.3 ~JournalTopicAck	1743
6.365.2 Member Function Documentation	1743
6.365.2.1 cloneDataStructure	1743
6.365.2.2 copyDataStructure	1743

6.365.2.3 equals	1743
6.365.2.4 getClientId	1744
6.365.2.5 getClientId	1744
6.365.2.6 getDataStructureType	1744
6.365.2.7 getDestination	1745
6.365.2.8 getDestination	1745
6.365.2.9 getMessageId	1745
6.365.2.10getMessageId	1745
6.365.2.11getMessageSequenceId	1745
6.365.2.12getSubscriptionName	1745
6.365.2.13getSubscriptionName	1745
6.365.2.14getTransactionId	1745
6.365.2.15getTransactionId	1745
6.365.2.16operator=	1745
6.365.2.17setClientId	1745
6.365.2.18setDestination	1745
6.365.2.19setMessageId	1745
6.365.2.20setMessageSequenceId	1745
6.365.2.21setSubscriptionName	1745
6.365.2.22setTransactionId	1745
6.365.2.23toString	1745
6.365.3 Field Documentation	1746
6.365.3.1 clientId	1746
6.365.3.2 destination	1746
6.365.3.3 ID_JOURNALTOPICACK	1746
6.365.3.4 messageId	1746
6.365.3.5 messageSequenceId	1746
6.365.3.6 subscriptionName	1746
6.365.3.7 transactionId	1746
6.366activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller Class	
Reference	1746
6.366.1 Detailed Description	1747
6.366.2 Constructor & Destructor Documentation	1748
6.366.2.1 JournalTopicAckMarshaller	1748
6.366.2.2 ~JournalTopicAckMarshaller	1748
6.366.3 Member Function Documentation	1748
6.366.3.1 createObject	1748

6.366.3.2	getDataStructureType	1748
6.366.3.3	looseMarshal	1748
6.366.3.4	looseUnmarshal	1749
6.366.3.5	tightMarshal1	1749
6.366.3.6	tightMarshal2	1749
6.366.3.7	tightUnmarshal	1750
6.367	activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller Class	
	Reference	1750
6.367.1	Detailed Description	1751
6.367.2	Constructor & Destructor Documentation	1752
6.367.2.1	JournalTopicAckMarshaller	1752
6.367.2.2	~JournalTopicAckMarshaller	1752
6.367.3	Member Function Documentation	1752
6.367.3.1	createObject	1752
6.367.3.2	getDataStructureType	1752
6.367.3.3	looseMarshal	1752
6.367.3.4	looseUnmarshal	1753
6.367.3.5	tightMarshal1	1753
6.367.3.6	tightMarshal2	1753
6.367.3.7	tightUnmarshal	1754
6.368	activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller Class	
	Reference	1754
6.368.1	Detailed Description	1755
6.368.2	Constructor & Destructor Documentation	1756
6.368.2.1	JournalTopicAckMarshaller	1756
6.368.2.2	~JournalTopicAckMarshaller	1756
6.368.3	Member Function Documentation	1756
6.368.3.1	createObject	1756
6.368.3.2	getDataStructureType	1756
6.368.3.3	looseMarshal	1756
6.368.3.4	looseUnmarshal	1757
6.368.3.5	tightMarshal1	1757
6.368.3.6	tightMarshal2	1757
6.368.3.7	tightUnmarshal	1758
6.369	activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller Class	
	Reference	1758
6.369.1	Detailed Description	1759

6.369.2 Constructor & Destructor Documentation	1760
6.369.2.1 JournalTopicAckMarshaller	1760
6.369.2.2 ~JournalTopicAckMarshaller	1760
6.369.3 Member Function Documentation	1760
6.369.3.1 createObject	1760
6.369.3.2 getDataStructureType	1760
6.369.3.3 looseMarshal	1760
6.369.3.4 looseUnmarshal	1761
6.369.3.5 tightMarshal1	1761
6.369.3.6 tightMarshal2	1761
6.369.3.7 tightUnmarshal	1762
6.370activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller Class Reference	1762
6.370.1 Detailed Description	1763
6.370.2 Constructor & Destructor Documentation	1764
6.370.2.1 JournalTopicAckMarshaller	1764
6.370.2.2 ~JournalTopicAckMarshaller	1764
6.370.3 Member Function Documentation	1764
6.370.3.1 createObject	1764
6.370.3.2 getDataStructureType	1764
6.370.3.3 looseMarshal	1764
6.370.3.4 looseUnmarshal	1765
6.370.3.5 tightMarshal1	1765
6.370.3.6 tightMarshal2	1765
6.370.3.7 tightUnmarshal	1766
6.371activemq::commands::JournalTrace Class Reference	1766
6.371.1 Constructor & Destructor Documentation	1767
6.371.1.1 JournalTrace	1767
6.371.1.2 JournalTrace	1767
6.371.1.3 ~JournalTrace	1767
6.371.2 Member Function Documentation	1767
6.371.2.1 cloneDataStructure	1767
6.371.2.2 copyDataStructure	1768
6.371.2.3 equals	1768
6.371.2.4 getDataStructureType	1768
6.371.2.5 getMessage	1769
6.371.2.6 getMessage	1769

6.371.2.7 operator=	1769
6.371.2.8 setMessage	1769
6.371.2.9 toString	1769
6.371.3 Field Documentation	1769
6.371.3.1 ID_JOURNALTRACE	1769
6.371.3.2 message	1769
6.372activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller Class Reference	1769
6.372.1 Detailed Description	1770
6.372.2 Constructor & Destructor Documentation	1771
6.372.2.1 JournalTraceMarshaller	1771
6.372.2.2 ~JournalTraceMarshaller	1771
6.372.3 Member Function Documentation	1771
6.372.3.1 createObject	1771
6.372.3.2 getDataStructureType	1771
6.372.3.3 looseMarshal	1771
6.372.3.4 looseUnmarshal	1772
6.372.3.5 tightMarshal1	1772
6.372.3.6 tightMarshal2	1772
6.372.3.7 tightUnmarshal	1773
6.373activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller Class Reference	1773
6.373.1 Detailed Description	1774
6.373.2 Constructor & Destructor Documentation	1775
6.373.2.1 JournalTraceMarshaller	1775
6.373.2.2 ~JournalTraceMarshaller	1775
6.373.3 Member Function Documentation	1775
6.373.3.1 createObject	1775
6.373.3.2 getDataStructureType	1775
6.373.3.3 looseMarshal	1775
6.373.3.4 looseUnmarshal	1776
6.373.3.5 tightMarshal1	1776
6.373.3.6 tightMarshal2	1776
6.373.3.7 tightUnmarshal	1777
6.374activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller Class Reference	1777
6.374.1 Detailed Description	1778

6.374.2 Constructor & Destructor Documentation	1779
6.374.2.1 JournalTraceMarshaller	1779
6.374.2.2 ~JournalTraceMarshaller	1779
6.374.3 Member Function Documentation	1779
6.374.3.1 createObject	1779
6.374.3.2 getDataStructureType	1779
6.374.3.3 looseMarshal	1779
6.374.3.4 looseUnmarshal	1780
6.374.3.5 tightMarshal1	1780
6.374.3.6 tightMarshal2	1780
6.374.3.7 tightUnmarshal	1781
6.375activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller Class Reference	1781
6.375.1 Detailed Description	1782
6.375.2 Constructor & Destructor Documentation	1783
6.375.2.1 JournalTraceMarshaller	1783
6.375.2.2 ~JournalTraceMarshaller	1783
6.375.3 Member Function Documentation	1783
6.375.3.1 createObject	1783
6.375.3.2 getDataStructureType	1783
6.375.3.3 looseMarshal	1783
6.375.3.4 looseUnmarshal	1784
6.375.3.5 tightMarshal1	1784
6.375.3.6 tightMarshal2	1784
6.375.3.7 tightUnmarshal	1785
6.376activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller Class Reference	1785
6.376.1 Detailed Description	1786
6.376.2 Constructor & Destructor Documentation	1787
6.376.2.1 JournalTraceMarshaller	1787
6.376.2.2 ~JournalTraceMarshaller	1787
6.376.3 Member Function Documentation	1787
6.376.3.1 createObject	1787
6.376.3.2 getDataStructureType	1787
6.376.3.3 looseMarshal	1787
6.376.3.4 looseUnmarshal	1788
6.376.3.5 tightMarshal1	1788

6.376.3.6 tightMarshal2	1788
6.376.3.7 tightUnmarshal	1789
6.377activemq::commands::JournalTransaction Class Reference	1789
6.377.1 Constructor & Destructor Documentation	1791
6.377.1.1 JournalTransaction	1791
6.377.1.2 JournalTransaction	1791
6.377.1.3 ~JournalTransaction	1791
6.377.2 Member Function Documentation	1791
6.377.2.1 cloneDataStructure	1791
6.377.2.2 copyDataStructure	1791
6.377.2.3 equals	1791
6.377.2.4 getDataStructureType	1792
6.377.2.5 getTransactionId	1792
6.377.2.6 getTransactionId	1792
6.377.2.7 getType	1792
6.377.2.8 getWasPrepared	1792
6.377.2.9 operator=	1792
6.377.2.10setTransactionId	1792
6.377.2.11setType	1792
6.377.2.12setWasPrepared	1792
6.377.2.13toString	1792
6.377.3 Field Documentation	1793
6.377.3.1 ID_ JOURNALTRANSACTION	1793
6.377.3.2 transactionId	1793
6.377.3.3 type	1793
6.377.3.4 wasPrepared	1793
6.378activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller Class Reference	1793
6.378.1 Detailed Description	1794
6.378.2 Constructor & Destructor Documentation	1794
6.378.2.1 JournalTransactionMarshaller	1794
6.378.2.2 ~JournalTransactionMarshaller	1794
6.378.3 Member Function Documentation	1794
6.378.3.1 createObject	1794
6.378.3.2 getDataStructureType	1794
6.378.3.3 looseMarshal	1795
6.378.3.4 looseUnmarshal	1795

6.378.3.5 tightMarshal1	1795
6.378.3.6 tightMarshal2	1796
6.378.3.7 tightUnmarshal	1796
6.379activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller Class	
Reference	1797
6.379.1 Detailed Description	1798
6.379.2 Constructor & Destructor Documentation	1798
6.379.2.1 JournalTransactionMarshaller	1798
6.379.2.2 ~JournalTransactionMarshaller	1798
6.379.3 Member Function Documentation	1798
6.379.3.1 createObject	1798
6.379.3.2 getDataStructureType	1798
6.379.3.3 looseMarshal	1799
6.379.3.4 looseUnmarshal	1799
6.379.3.5 tightMarshal1	1799
6.379.3.6 tightMarshal2	1800
6.379.3.7 tightUnmarshal	1800
6.380activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller Class	
Reference	1801
6.380.1 Detailed Description	1802
6.380.2 Constructor & Destructor Documentation	1802
6.380.2.1 JournalTransactionMarshaller	1802
6.380.2.2 ~JournalTransactionMarshaller	1802
6.380.3 Member Function Documentation	1802
6.380.3.1 createObject	1802
6.380.3.2 getDataStructureType	1802
6.380.3.3 looseMarshal	1802
6.380.3.4 looseUnmarshal	1803
6.380.3.5 tightMarshal1	1803
6.380.3.6 tightMarshal2	1804
6.380.3.7 tightUnmarshal	1804
6.381activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller Class	
Reference	1804
6.381.1 Detailed Description	1805
6.381.2 Constructor & Destructor Documentation	1806
6.381.2.1 JournalTransactionMarshaller	1806
6.381.2.2 ~JournalTransactionMarshaller	1806

6.381.3 Member Function Documentation	1806
6.381.3.1 createObject	1806
6.381.3.2 getDataStructureType	1806
6.381.3.3 looseMarshal	1806
6.381.3.4 looseUnmarshal	1807
6.381.3.5 tightMarshal1	1807
6.381.3.6 tightMarshal2	1807
6.381.3.7 tightUnmarshal	1808
6.382activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller Class Reference	1808
6.382.1 Detailed Description	1809
6.382.2 Constructor & Destructor Documentation	1810
6.382.2.1 JournalTransactionMarshaller	1810
6.382.2.2 ~JournalTransactionMarshaller	1810
6.382.3 Member Function Documentation	1810
6.382.3.1 createObject	1810
6.382.3.2 getDataStructureType	1810
6.382.3.3 looseMarshal	1810
6.382.3.4 looseUnmarshal	1811
6.382.3.5 tightMarshal1	1811
6.382.3.6 tightMarshal2	1811
6.382.3.7 tightUnmarshal	1812
6.383activemq::commands::KeepAliveInfo Class Reference	1812
6.383.1 Constructor & Destructor Documentation	1813
6.383.1.1 KeepAliveInfo	1813
6.383.1.2 KeepAliveInfo	1813
6.383.1.3 ~KeepAliveInfo	1813
6.383.2 Member Function Documentation	1813
6.383.2.1 cloneDataStructure	1813
6.383.2.2 copyDataStructure	1814
6.383.2.3 equals	1814
6.383.2.4 getDataStructureType	1814
6.383.2.5 isKeepAliveInfo	1814
6.383.2.6 operator=	1815
6.383.2.7 toString	1815
6.383.2.8 visit	1815
6.383.3 Field Documentation	1815

6.383.3.1 ID_KEEPAALIVEINFO	1815
6.384activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller Class Reference	1815
6.384.1 Detailed Description	1816
6.384.2 Constructor & Destructor Documentation	1817
6.384.2.1 KeepAliveInfoMarshaller	1817
6.384.2.2 ~KeepAliveInfoMarshaller	1817
6.384.3 Member Function Documentation	1817
6.384.3.1 createObject	1817
6.384.3.2 getDataStructureType	1817
6.384.3.3 looseMarshal	1817
6.384.3.4 looseUnmarshal	1818
6.384.3.5 tightMarshal1	1818
6.384.3.6 tightMarshal2	1818
6.384.3.7 tightUnmarshal	1819
6.385activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller Class Reference	1819
6.385.1 Detailed Description	1820
6.385.2 Constructor & Destructor Documentation	1821
6.385.2.1 KeepAliveInfoMarshaller	1821
6.385.2.2 ~KeepAliveInfoMarshaller	1821
6.385.3 Member Function Documentation	1821
6.385.3.1 createObject	1821
6.385.3.2 getDataStructureType	1821
6.385.3.3 looseMarshal	1821
6.385.3.4 looseUnmarshal	1822
6.385.3.5 tightMarshal1	1822
6.385.3.6 tightMarshal2	1822
6.385.3.7 tightUnmarshal	1823
6.386activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller Class Reference	1823
6.386.1 Detailed Description	1824
6.386.2 Constructor & Destructor Documentation	1825
6.386.2.1 KeepAliveInfoMarshaller	1825
6.386.2.2 ~KeepAliveInfoMarshaller	1825
6.386.3 Member Function Documentation	1825
6.386.3.1 createObject	1825

6.386.3.2	getDataStructureType	1825
6.386.3.3	looseMarshal	1825
6.386.3.4	looseUnmarshal	1826
6.386.3.5	tightMarshal1	1826
6.386.3.6	tightMarshal2	1826
6.386.3.7	tightUnmarshal	1827
6.387	activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller Class Reference	1827
6.387.1	Detailed Description	1828
6.387.2	Constructor & Destructor Documentation	1829
6.387.2.1	KeepAliveInfoMarshaller	1829
6.387.2.2	~KeepAliveInfoMarshaller	1829
6.387.3	Member Function Documentation	1829
6.387.3.1	createObject	1829
6.387.3.2	getDataStructureType	1829
6.387.3.3	looseMarshal	1829
6.387.3.4	looseUnmarshal	1830
6.387.3.5	tightMarshal1	1830
6.387.3.6	tightMarshal2	1830
6.387.3.7	tightUnmarshal	1831
6.388	activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller Class Reference	1831
6.388.1	Detailed Description	1832
6.388.2	Constructor & Destructor Documentation	1833
6.388.2.1	KeepAliveInfoMarshaller	1833
6.388.2.2	~KeepAliveInfoMarshaller	1833
6.388.3	Member Function Documentation	1833
6.388.3.1	createObject	1833
6.388.3.2	getDataStructureType	1833
6.388.3.3	looseMarshal	1833
6.388.3.4	looseUnmarshal	1834
6.388.3.5	tightMarshal1	1834
6.388.3.6	tightMarshal2	1834
6.388.3.7	tightUnmarshal	1835
6.389	decaf::security::Key Class Reference	1835
6.389.1	Detailed Description	1836
6.389.2	Constructor & Destructor Documentation	1837

6.389.2.1 ~Key	1837
6.389.3 Member Function Documentation	1837
6.389.3.1 getAlgorithm	1837
6.389.3.2 getEncoded	1837
6.389.3.3 getFormat	1837
6.390decaf::security::KeyException Class Reference	1837
6.390.1 Constructor & Destructor Documentation	1838
6.390.1.1 KeyException	1838
6.390.1.2 KeyException	1838
6.390.1.3 KeyException	1838
6.390.1.4 KeyException	1839
6.390.1.5 KeyException	1839
6.390.1.6 KeyException	1839
6.390.1.7 ~KeyException	1839
6.390.2 Member Function Documentation	1839
6.390.2.1 clone	1839
6.391activemq::commands::LastPartialCommand Class Reference	1840
6.391.1 Constructor & Destructor Documentation	1841
6.391.1.1 LastPartialCommand	1841
6.391.1.2 LastPartialCommand	1841
6.391.1.3 ~LastPartialCommand	1841
6.391.2 Member Function Documentation	1841
6.391.2.1 cloneDataStructure	1841
6.391.2.2 copyDataStructure	1841
6.391.2.3 equals	1841
6.391.2.4 getDataStructureType	1842
6.391.2.5 operator=	1842
6.391.2.6 toString	1842
6.391.3 Field Documentation	1842
6.391.3.1 ID_LASTPARTIALCOMMAND	1842
6.392activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller Class Reference	1842
6.392.1 Detailed Description	1843
6.392.2 Constructor & Destructor Documentation	1844
6.392.2.1 LastPartialCommandMarshaller	1844
6.392.2.2 ~LastPartialCommandMarshaller	1844
6.392.3 Member Function Documentation	1844

6.392.3.1 createObject	1844
6.392.3.2 getDataStructureType	1844
6.392.3.3 looseMarshal	1844
6.392.3.4 looseUnmarshal	1845
6.392.3.5 tightMarshal1	1845
6.392.3.6 tightMarshal2	1846
6.392.3.7 tightUnmarshal	1846
6.393activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller	
Class Reference	1846
6.393.1 Detailed Description	1847
6.393.2 Constructor & Destructor Documentation	1848
6.393.2.1 LastPartialCommandMarshaller	1848
6.393.2.2 ~LastPartialCommandMarshaller	1848
6.393.3 Member Function Documentation	1848
6.393.3.1 createObject	1848
6.393.3.2 getDataStructureType	1848
6.393.3.3 looseMarshal	1848
6.393.3.4 looseUnmarshal	1849
6.393.3.5 tightMarshal1	1849
6.393.3.6 tightMarshal2	1850
6.393.3.7 tightUnmarshal	1850
6.394activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller	
Class Reference	1850
6.394.1 Detailed Description	1851
6.394.2 Constructor & Destructor Documentation	1852
6.394.2.1 LastPartialCommandMarshaller	1852
6.394.2.2 ~LastPartialCommandMarshaller	1852
6.394.3 Member Function Documentation	1852
6.394.3.1 createObject	1852
6.394.3.2 getDataStructureType	1852
6.394.3.3 looseMarshal	1852
6.394.3.4 looseUnmarshal	1853
6.394.3.5 tightMarshal1	1853
6.394.3.6 tightMarshal2	1854
6.394.3.7 tightUnmarshal	1854
6.395activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller	
Class Reference	1854

6.395.1 Detailed Description	1855
6.395.2 Constructor & Destructor Documentation	1856
6.395.2.1 LastPartialCommandMarshaller	1856
6.395.2.2 ~LastPartialCommandMarshaller	1856
6.395.3 Member Function Documentation	1856
6.395.3.1 createObject	1856
6.395.3.2 getDataStructureType	1856
6.395.3.3 looseMarshal	1856
6.395.3.4 looseUnmarshal	1857
6.395.3.5 tightMarshal1	1857
6.395.3.6 tightMarshal2	1858
6.395.3.7 tightUnmarshal	1858
6.396activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller	
Class Reference	1858
6.396.1 Detailed Description	1859
6.396.2 Constructor & Destructor Documentation	1860
6.396.2.1 LastPartialCommandMarshaller	1860
6.396.2.2 ~LastPartialCommandMarshaller	1860
6.396.3 Member Function Documentation	1860
6.396.3.1 createObject	1860
6.396.3.2 getDataStructureType	1860
6.396.3.3 looseMarshal	1860
6.396.3.4 looseUnmarshal	1861
6.396.3.5 tightMarshal1	1861
6.396.3.6 tightMarshal2	1862
6.396.3.7 tightUnmarshal	1862
6.397decaf::util::comparators::Less< E > Class Template Reference	1862
6.397.1 Detailed Description	1863
6.397.2 Constructor & Destructor Documentation	1863
6.397.2.1 Less	1863
6.397.2.2 ~Less	1863
6.397.3 Member Function Documentation	1863
6.397.3.1 compare	1863
6.397.3.2 operator()	1864
6.398std::less< decaf::lang::Pointer< T > > Struct Template Reference	1864
6.398.1 Detailed Description	1865
6.398.2 Member Function Documentation	1865

6.398.2.1 operator()	1865
6.399decaf::util::List< E > Class Template Reference	1865
6.399.1 Detailed Description	1866
6.399.2 Constructor & Destructor Documentation	1867
6.399.2.1 ~List	1867
6.399.3 Member Function Documentation	1867
6.399.3.1 add	1867
6.399.3.2 addAll	1867
6.399.3.3 get	1868
6.399.3.4 indexOf	1868
6.399.3.5 lastIndexOf	1869
6.399.3.6 listIterator	1869
6.399.3.7 listIterator	1869
6.399.3.8 listIterator	1869
6.399.3.9 listIterator	1870
6.399.3.10 remove	1870
6.399.3.11 set	1871
6.400decaf::util::ListIterator< E > Class Template Reference	1871
6.400.1 Detailed Description	1872
6.400.2 Constructor & Destructor Documentation	1872
6.400.2.1 ~ListIterator	1872
6.400.3 Member Function Documentation	1872
6.400.3.1 add	1872
6.400.3.2 hasPrevious	1873
6.400.3.3 nextIndex	1873
6.400.3.4 previous	1873
6.400.3.5 previousIndex	1873
6.400.3.6 set	1874
6.401activemq::commands::LocalTransactionId Class Reference	1874
6.401.1 Member Typedef Documentation	1876
6.401.1.1 COMPARATOR	1876
6.401.2 Constructor & Destructor Documentation	1876
6.401.2.1 LocalTransactionId	1876
6.401.2.2 LocalTransactionId	1876
6.401.2.3 ~LocalTransactionId	1876
6.401.3 Member Function Documentation	1876

6.401.3.1 cloneDataStructure	1876
6.401.3.2 compareTo	1876
6.401.3.3 copyDataStructure	1876
6.401.3.4 equals	1876
6.401.3.5 equals	1877
6.401.3.6 getConnectionId	1877
6.401.3.7 getConnectionId	1877
6.401.3.8 getDataStructureType	1877
6.401.3.9 getValue	1877
6.401.3.10 operator<	1877
6.401.3.11 operator=	1877
6.401.3.12 operator==	1877
6.401.3.13 setConnectionId	1877
6.401.3.14 setValue	1877
6.401.3.15 toString	1877
6.401.4 Field Documentation	1878
6.401.4.1 connectionId	1878
6.401.4.2 ID_LOCALTRANSACTIONID	1878
6.401.4.3 value	1878
6.402activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller Class	
Reference	1878
6.402.1 Detailed Description	1879
6.402.2 Constructor & Destructor Documentation	1879
6.402.2.1 LocalTransactionIdMarshaller	1879
6.402.2.2 ~LocalTransactionIdMarshaller	1879
6.402.3 Member Function Documentation	1879
6.402.3.1 createObject	1879
6.402.3.2 getDataStructureType	1879
6.402.3.3 looseMarshal	1880
6.402.3.4 looseUnmarshal	1880
6.402.3.5 tightMarshal1	1880
6.402.3.6 tightMarshal2	1881
6.402.3.7 tightUnmarshal	1881
6.403activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller Class	
Reference	1882
6.403.1 Detailed Description	1883
6.403.2 Constructor & Destructor Documentation	1883

6.403.2.1 LocalTransactionIdMarshaller	1883
6.403.2.2 ~LocalTransactionIdMarshaller	1883
6.403.3 Member Function Documentation	1883
6.403.3.1 createObject	1883
6.403.3.2 getDataStructureType	1883
6.403.3.3 looseMarshal	1884
6.403.3.4 looseUnmarshal	1884
6.403.3.5 tightMarshal1	1884
6.403.3.6 tightMarshal2	1885
6.403.3.7 tightUnmarshal	1885
6.404activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller Class	
Reference	1886
6.404.1 Detailed Description	1887
6.404.2 Constructor & Destructor Documentation	1887
6.404.2.1 LocalTransactionIdMarshaller	1887
6.404.2.2 ~LocalTransactionIdMarshaller	1887
6.404.3 Member Function Documentation	1887
6.404.3.1 createObject	1887
6.404.3.2 getDataStructureType	1887
6.404.3.3 looseMarshal	1888
6.404.3.4 looseUnmarshal	1888
6.404.3.5 tightMarshal1	1888
6.404.3.6 tightMarshal2	1889
6.404.3.7 tightUnmarshal	1889
6.405activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller Class	
Reference	1890
6.405.1 Detailed Description	1891
6.405.2 Constructor & Destructor Documentation	1891
6.405.2.1 LocalTransactionIdMarshaller	1891
6.405.2.2 ~LocalTransactionIdMarshaller	1891
6.405.3 Member Function Documentation	1891
6.405.3.1 createObject	1891
6.405.3.2 getDataStructureType	1891
6.405.3.3 looseMarshal	1892
6.405.3.4 looseUnmarshal	1892
6.405.3.5 tightMarshal1	1892
6.405.3.6 tightMarshal2	1893

6.405.3.7 tightUnmarshal	1893
6.406activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller Class Reference	1894
6.406.1 Detailed Description	1895
6.406.2 Constructor & Destructor Documentation	1895
6.406.2.1 LocalTransactionIdMarshaller	1895
6.406.2.2 ~LocalTransactionIdMarshaller	1895
6.406.3 Member Function Documentation	1895
6.406.3.1 createObject	1895
6.406.3.2 getDataStructureType	1895
6.406.3.3 looseMarshal	1896
6.406.3.4 looseUnmarshal	1896
6.406.3.5 tightMarshal1	1896
6.406.3.6 tightMarshal2	1897
6.406.3.7 tightUnmarshal	1897
6.407decaf::util::concurrent::locks::Lock Class Reference	1898
6.407.1 Detailed Description	1899
6.407.2 Constructor & Destructor Documentation	1900
6.407.2.1 ~Lock	1900
6.407.3 Member Function Documentation	1900
6.407.3.1 lock	1900
6.407.3.2 lockInterruptibly	1900
6.407.3.3 newCondition	1901
6.407.3.4 tryLock	1901
6.407.3.5 tryLock	1902
6.407.3.6 unlock	1903
6.408decaf::util::concurrent::Lock Class Reference	1903
6.408.1 Detailed Description	1904
6.408.2 Constructor & Destructor Documentation	1904
6.408.2.1 Lock	1904
6.408.2.2 ~Lock	1904
6.408.3 Member Function Documentation	1904
6.408.3.1 isLocked	1904
6.408.3.2 lock	1904
6.408.3.3 unlock	1905
6.409decaf::util::concurrent::locks::LockSupport Class Reference	1905
6.409.1 Detailed Description	1905

6.409.2 Constructor & Destructor Documentation	1906
6.409.2.1 ~LockSupport	1906
6.409.3 Member Function Documentation	1906
6.409.3.1 park	1906
6.409.3.2 parkNanos	1907
6.409.3.3 parkUntil	1907
6.409.3.4 unpark	1907
6.410decaf::util::logging::Logger Class Reference	1908
6.410.1 Constructor & Destructor Documentation	1910
6.410.1.1 Logger	1910
6.410.1.2 ~Logger	1910
6.410.2 Member Function Documentation	1910
6.410.2.1 addHandler	1910
6.410.2.2 debug	1910
6.410.2.3 entry	1911
6.410.2.4 error	1911
6.410.2.5 exit	1911
6.410.2.6 fatal	1911
6.410.2.7 getAnonymousLogger	1912
6.410.2.8 getFilter	1912
6.410.2.9 getLevel	1912
6.410.2.10getLogger	1912
6.410.2.11getName	1913
6.410.2.12getUseParentHandlers	1913
6.410.2.13info	1913
6.410.2.14sLoggable	1913
6.410.2.15log	1914
6.410.2.16og	1914
6.410.2.17og	1914
6.410.2.18og	1915
6.410.2.19removeHandler	1915
6.410.2.20setFilter	1915
6.410.2.21setLevel	1916
6.410.2.22setUseParentHandlers	1916
6.410.2.23warn	1916
6.411decaf::util::logging::LoggerHierarchy Class Reference	1916

6.411.1 Constructor & Destructor Documentation	1917
6.411.1.1 LoggerHierarchy	1917
6.411.1.2 ~LoggerHierarchy	1917
6.412activemq::io::LoggingInputStream Class Reference	1917
6.412.1 Constructor & Destructor Documentation	1917
6.412.1.1 LoggingInputStream	1917
6.412.1.2 ~LoggingInputStream	1918
6.412.2 Member Function Documentation	1918
6.412.2.1 read	1918
6.412.2.2 read	1918
6.413activemq::io::LoggingOutputStream Class Reference	1919
6.413.1 Detailed Description	1919
6.413.2 Constructor & Destructor Documentation	1919
6.413.2.1 LoggingOutputStream	1919
6.413.2.2 ~LoggingOutputStream	1919
6.413.3 Member Function Documentation	1919
6.413.3.1 write	1919
6.413.3.2 write	1920
6.414activemq::transport::logging::LoggingTransport Class Reference	1920
6.414.1 Detailed Description	1921
6.414.2 Constructor & Destructor Documentation	1921
6.414.2.1 LoggingTransport	1921
6.414.2.2 ~LoggingTransport	1921
6.414.3 Member Function Documentation	1921
6.414.3.1 onCommand	1921
6.414.3.2 oneway	1922
6.414.3.3 request	1922
6.414.3.4 request	1922
6.415decaf::util::logging::LogManager Class Reference	1923
6.415.1 Detailed Description	1924
6.415.2 Constructor & Destructor Documentation	1925
6.415.2.1 ~LogManager	1925
6.415.2.2 LogManager	1925
6.415.2.3 LogManager	1925
6.415.3 Member Function Documentation	1925
6.415.3.1 addPropertyChangeListener	1925

6.415.3.2	destroy	1926
6.415.3.3	getInstance	1926
6.415.3.4	getLogger	1926
6.415.3.5	getLoggerNames	1926
6.415.3.6	getProperties	1926
6.415.3.7	getProperty	1927
6.415.3.8	operator=	1927
6.415.3.9	removePropertyChangeListener	1927
6.415.3.10	returnInstance	1927
6.415.3.11	setProperties	1927
6.416	decaf::util::logging::LogRecord Class Reference	1928
6.416.1	Constructor & Destructor Documentation	1929
6.416.1.1	LogRecord	1929
6.416.1.2	~LogRecord	1929
6.416.2	Member Function Documentation	1929
6.416.2.1	getLevel	1929
6.416.2.2	getLoggerName	1929
6.416.2.3	getMessage	1929
6.416.2.4	getSourceFile	1929
6.416.2.5	getSourceFunction	1930
6.416.2.6	getSourceLine	1930
6.416.2.7	getTimestamp	1930
6.416.2.8	getTreadId	1930
6.416.2.9	setLevel	1930
6.416.2.10	setLoggerName	1931
6.416.2.11	setMessage	1931
6.416.2.12	setSourceFile	1931
6.416.2.13	setSourceFunction	1931
6.416.2.14	setSourceLine	1931
6.416.2.15	setTimestamp	1931
6.416.2.16	setTreadId	1932
6.417	decaf::util::logging::LogWriter Class Reference	1932
6.417.1	Constructor & Destructor Documentation	1933
6.417.1.1	LogWriter	1933
6.417.1.2	~LogWriter	1933
6.417.2	Member Function Documentation	1933

6.417.2.1	destroy	1933
6.417.2.2	getInstance	1933
6.417.2.3	log	1933
6.417.2.4	log	1933
6.417.2.5	returnInstance	1933
6.418	decaf::lang::Long Class Reference	1934
6.418.1	Constructor & Destructor Documentation	1937
6.418.1.1	Long	1937
6.418.1.2	Long	1937
6.418.1.3	~Long	1937
6.418.2	Member Function Documentation	1937
6.418.2.1	bitCount	1937
6.418.2.2	byteValue	1937
6.418.2.3	compareTo	1937
6.418.2.4	compareTo	1938
6.418.2.5	decode	1938
6.418.2.6	doubleValue	1939
6.418.2.7	equals	1939
6.418.2.8	equals	1939
6.418.2.9	float Value	1939
6.418.2.10	highestOneBit	1939
6.418.2.11	int Value	1940
6.418.2.12	long Value	1940
6.418.2.13	lowestOneBit	1940
6.418.2.14	numberOfLeadingZeros	1940
6.418.2.15	numberOfTrailingZeros	1941
6.418.2.16	operator<	1941
6.418.2.17	operator<	1941
6.418.2.18	operator==	1942
6.418.2.19	operator==	1942
6.418.2.20	parseLong	1942
6.418.2.21	parseLong	1943
6.418.2.22	reverse	1943
6.418.2.23	reverseBytes	1943
6.418.2.24	rotateLeft	1944
6.418.2.25	rotateRight	1944

6.418.2.26	shortValue	1944
6.418.2.27	signum	1945
6.418.2.28	oBinaryString	1945
6.418.2.29	oHexString	1945
6.418.2.30	oOctalString	1946
6.418.2.31	toString	1946
6.418.2.32	oString	1946
6.418.2.33	oString	1946
6.418.2.34	valueOf	1946
6.418.2.35	valueOf	1947
6.418.2.36	valueOf	1947
6.418.3	Field Documentation	1947
6.418.3.1	MAX_VALUE	1947
6.418.3.2	MIN_VALUE	1947
6.418.3.3	SIZE	1948
6.419	decaf::nio::LongArrayBuffer Class Reference	1948
6.419.1	Constructor & Destructor Documentation	1949
6.419.1.1	LongArrayBuffer	1949
6.419.1.2	LongArrayBuffer	1950
6.419.1.3	LongArrayBuffer	1950
6.419.1.4	LongArrayBuffer	1950
6.419.1.5	~LongArrayBuffer	1951
6.419.2	Member Function Documentation	1951
6.419.2.1	array	1951
6.419.2.2	arrayOffset	1951
6.419.2.3	asReadOnlyBuffer	1951
6.419.2.4	compact	1952
6.419.2.5	duplicate	1952
6.419.2.6	get	1953
6.419.2.7	get	1953
6.419.2.8	hasArray	1953
6.419.2.9	isReadOnly	1954
6.419.2.10	put	1954
6.419.2.11	put	1954
6.419.2.12	setReadOnly	1955
6.419.2.13	slice	1955

6.420decaf::nio::LongBuffer Class Reference	1955
6.420.1 Detailed Description	1957
6.420.2 Constructor & Destructor Documentation	1958
6.420.2.1 LongBuffer	1958
6.420.2.2 ~LongBuffer	1958
6.420.3 Member Function Documentation	1958
6.420.3.1 allocate	1958
6.420.3.2 array	1958
6.420.3.3 arrayOffset	1959
6.420.3.4 asReadOnlyBuffer	1959
6.420.3.5 compact	1959
6.420.3.6 compareTo	1960
6.420.3.7 duplicate	1960
6.420.3.8 equals	1960
6.420.3.9 get	1961
6.420.3.10get	1961
6.420.3.11get	1961
6.420.3.12get	1962
6.420.3.13hasArray	1962
6.420.3.14operator<	1962
6.420.3.15operator==	1963
6.420.3.16put	1963
6.420.3.17put	1963
6.420.3.18put	1964
6.420.3.19put	1964
6.420.3.20put	1965
6.420.3.21slice	1965
6.420.3.22toString	1966
6.420.3.23wrap	1966
6.420.3.24wrap	1966
6.421activemq::util::LongSequenceGenerator Class Reference	1967
6.421.1 Detailed Description	1967
6.421.2 Constructor & Destructor Documentation	1967
6.421.2.1 LongSequenceGenerator	1967
6.421.2.2 ~LongSequenceGenerator	1967
6.421.3 Member Function Documentation	1967

6.421.3.1	getLastSequenceId	1967
6.421.3.2	getNextSequenceId	1967
6.422	decaf::net::MalformedURLException Class Reference	1968
6.422.1	Constructor & Destructor Documentation	1968
6.422.1.1	MalformedURLException	1968
6.422.1.2	MalformedURLException	1968
6.422.1.3	MalformedURLException	1969
6.422.1.4	MalformedURLException	1969
6.422.1.5	MalformedURLException	1969
6.422.1.6	MalformedURLException	1969
6.422.1.7	~MalformedURLException	1970
6.422.2	Member Function Documentation	1970
6.422.2.1	clone	1970
6.423	decaf::util::Map< K, V, COMPARATOR > Class Template Reference	1970
6.423.1	Detailed Description	1971
6.423.2	Constructor & Destructor Documentation	1972
6.423.2.1	Map	1972
6.423.2.2	~Map	1972
6.423.3	Member Function Documentation	1972
6.423.3.1	clear	1972
6.423.3.2	containsKey	1973
6.423.3.3	containsValue	1973
6.423.3.4	copy	1974
6.423.3.5	equals	1974
6.423.3.6	get	1975
6.423.3.7	get	1976
6.423.3.8	isEmpty	1977
6.423.3.9	keySet	1977
6.423.3.10	put	1978
6.423.3.11	putAll	1979
6.423.3.12	remove	1980
6.423.3.13	size	1981
6.423.3.14	values	1981
6.424	cms::MapMessage Class Reference	1982
6.424.1	Detailed Description	1984
6.424.2	Constructor & Destructor Documentation	1985

6.424.2.1 ~MapMessage	1985
6.424.3 Member Function Documentation	1985
6.424.3.1 getBoolean	1985
6.424.3.2 getByte	1985
6.424.3.3 getBytes	1985
6.424.3.4 getChar	1986
6.424.3.5 getDouble	1986
6.424.3.6 getFloat	1986
6.424.3.7 getInt	1987
6.424.3.8 getLong	1987
6.424.3.9 getMapNames	1987
6.424.3.10 getShort	1987
6.424.3.11 getString	1988
6.424.3.12 itemExists	1988
6.424.3.13 setBoolean	1988
6.424.3.14 setByte	1989
6.424.3.15 setBytes	1989
6.424.3.16 setChar	1989
6.424.3.17 setDouble	1990
6.424.3.18 setFloat	1990
6.424.3.19 setInt	1990
6.424.3.20 setLong	1991
6.424.3.21 setShort	1991
6.424.3.22 setString	1991
6.425 decaf::util::logging::MarkBlockLogger Class Reference	1992
6.425.1 Detailed Description	1992
6.425.2 Constructor & Destructor Documentation	1992
6.425.2.1 MarkBlockLogger	1992
6.425.2.2 ~MarkBlockLogger	1992
6.426 activemq::wireformat::MarshalAware Class Reference	1992
6.426.1 Constructor & Destructor Documentation	1993
6.426.1.1 ~MarshalAware	1993
6.426.2 Member Function Documentation	1993
6.426.2.1 afterMarshal	1993
6.426.2.2 afterUnmarshal	1994
6.426.2.3 beforeMarshal	1994

6.426.2.4 beforeUnmarshal	1994
6.426.2.5 getMarshaledForm	1994
6.426.2.6 isMarshalAware	1995
6.426.2.7 setMarshaledForm	1995
6.427activemq::wireformat::openwire::marshal::v2::MarshallerFactory Class Reference . .	1995
6.427.1 Detailed Description	1995
6.427.2 Constructor & Destructor Documentation	1996
6.427.2.1 ~MarshallerFactory	1996
6.427.3 Member Function Documentation	1996
6.427.3.1 configure	1996
6.428activemq::wireformat::openwire::marshal::v4::MarshallerFactory Class Reference . .	1996
6.428.1 Detailed Description	1996
6.428.2 Constructor & Destructor Documentation	1996
6.428.2.1 ~MarshallerFactory	1996
6.428.3 Member Function Documentation	1996
6.428.3.1 configure	1996
6.429activemq::wireformat::openwire::marshal::v1::MarshallerFactory Class Reference . .	1997
6.429.1 Detailed Description	1997
6.429.2 Constructor & Destructor Documentation	1997
6.429.2.1 ~MarshallerFactory	1997
6.429.3 Member Function Documentation	1997
6.429.3.1 configure	1997
6.430activemq::wireformat::openwire::marshal::v3::MarshallerFactory Class Reference . .	1997
6.430.1 Detailed Description	1998
6.430.2 Constructor & Destructor Documentation	1998
6.430.2.1 ~MarshallerFactory	1998
6.430.3 Member Function Documentation	1998
6.430.3.1 configure	1998
6.431activemq::wireformat::openwire::marshal::v5::MarshallerFactory Class Reference . .	1998
6.431.1 Detailed Description	1998
6.431.2 Constructor & Destructor Documentation	1999
6.431.2.1 ~MarshallerFactory	1999
6.431.3 Member Function Documentation	1999
6.431.3.1 configure	1999
6.432decaf::lang::Math Class Reference	1999
6.432.1 Detailed Description	2001

6.432.2 Constructor & Destructor Documentation	2001
6.432.2.1 Math	2001
6.432.2.2 \sim Math	2001
6.432.3 Member Function Documentation	2001
6.432.3.1 abs	2001
6.432.3.2 abs	2001
6.432.3.3 abs	2002
6.432.3.4 abs	2002
6.432.3.5 ceil	2002
6.432.3.6 floor	2004
6.432.3.7 max	2004
6.432.3.8 max	2004
6.432.3.9 max	2005
6.432.3.10max	2005
6.432.3.11max	2005
6.432.3.12min	2006
6.432.3.13min	2006
6.432.3.14min	2006
6.432.3.15min	2007
6.432.3.16min	2007
6.432.3.17min	2007
6.432.3.18pow	2008
6.432.3.19random	2008
6.432.3.20round	2009
6.432.3.21round	2009
6.432.3.22signum	2010
6.432.3.23signum	2010
6.432.3.24qrt	2011
6.432.3.25toDegrees	2014
6.432.3.26toRadians	2014
6.432.4 Field Documentation	2014
6.432.4.1 E	2014
6.432.4.2 PI	2014
6.433activemq::util::MemoryUsage Class Reference	2014
6.433.1 Constructor & Destructor Documentation	2015
6.433.1.1 MemoryUsage	2015

6.433.1.2	MemoryUsage	2016
6.433.1.3	~MemoryUsage	2016
6.433.2	Member Function Documentation	2016
6.433.2.1	decreaseUsage	2016
6.433.2.2	enqueueUsage	2016
6.433.2.3	getLimit	2016
6.433.2.4	getUsage	2016
6.433.2.5	increaseUsage	2017
6.433.2.6	isFull	2017
6.433.2.7	setLimit	2017
6.433.2.8	setUsage	2017
6.433.2.9	waitForSpace	2017
6.433.2.10	waitForSpace	2017
6.434	activemq::commands::Message Class Reference	2018
6.434.1	Constructor & Destructor Documentation	2022
6.434.1.1	Message	2022
6.434.1.2	Message	2022
6.434.1.3	~Message	2022
6.434.2	Member Function Documentation	2022
6.434.2.1	afterUnmarshal	2022
6.434.2.2	beforeMarshal	2022
6.434.2.3	cloneDataStructure	2023
6.434.2.4	copyDataStructure	2023
6.434.2.5	equals	2023
6.434.2.6	getAckHandler	2024
6.434.2.7	getArrival	2025
6.434.2.8	getBrokerInTime	2025
6.434.2.9	getBrokerOutTime	2025
6.434.2.10	getBrokerPath	2025
6.434.2.11	getBrokerPath	2025
6.434.2.12	getCluster	2025
6.434.2.13	getCluster	2025
6.434.2.14	getContent	2025
6.434.2.15	getContent	2025
6.434.2.16	getCorrelationId	2025
6.434.2.17	getCorrelationId	2025

6.434.2.18	getDataStructure	2025
6.434.2.19	getDataStructure	2025
6.434.2.20	getDataStructureType	2025
6.434.2.21	getDestination	2027
6.434.2.22	getDestination	2027
6.434.2.23	getExpiration	2027
6.434.2.24	getGroupID	2027
6.434.2.25	getGroupID	2027
6.434.2.26	getGroupSequence	2027
6.434.2.27	getMarshaledProperties	2027
6.434.2.28	getMarshaledProperties	2027
6.434.2.29	getMessageId	2027
6.434.2.30	getMessageId	2027
6.434.2.31	getMessageProperties	2027
6.434.2.32	getMessageProperties	2027
6.434.2.33	getOriginalDestination	2028
6.434.2.34	getOriginalDestination	2028
6.434.2.35	getOriginalTransactionId	2028
6.434.2.36	getOriginalTransactionId	2028
6.434.2.37	getPriority	2028
6.434.2.38	getProducerId	2028
6.434.2.39	getProducerId	2028
6.434.2.40	getRedeliveryCounter	2028
6.434.2.41	getReplyTo	2028
6.434.2.42	getReplyTo	2028
6.434.2.43	getSize	2028
6.434.2.44	getTargetConsumerId	2029
6.434.2.45	getTargetConsumerId	2029
6.434.2.46	getTimestamp	2029
6.434.2.47	getTransactionId	2029
6.434.2.48	getTransactionId	2029
6.434.2.49	getType	2029
6.434.2.50	getType	2029
6.434.2.51	getUserID	2029
6.434.2.52	getUserID	2029
6.434.2.53	isCompressed	2029

6.434.2.54sDroppable	2029
6.434.2.55sExpired	2029
6.434.2.56sMarshalAware	2029
6.434.2.57sMessage	2030
6.434.2.58sPersistent	2030
6.434.2.59sReadOnlyBody	2030
6.434.2.60sReadOnlyProperties	2030
6.434.2.61sRecievedByDFBridge	2030
6.434.2.62nSend	2030
6.434.2.63operator=	2031
6.434.2.64setAckHandler	2031
6.434.2.65setArrival	2032
6.434.2.66setBrokerInTime	2032
6.434.2.67setBrokerOutTime	2032
6.434.2.68setBrokerPath	2032
6.434.2.69setCluster	2032
6.434.2.70setCompressed	2032
6.434.2.71setContent	2032
6.434.2.72setCorrelationId	2032
6.434.2.73setDataStructure	2032
6.434.2.74setDestination	2032
6.434.2.75setDroppable	2032
6.434.2.76setExpiration	2032
6.434.2.77setGroupID	2032
6.434.2.78setGroupSequence	2032
6.434.2.79setMarshaledProperties	2032
6.434.2.80setMessageId	2032
6.434.2.81setOriginalDestination	2032
6.434.2.82setOriginalTransactionId	2032
6.434.2.83setPersistent	2032
6.434.2.84setPriority	2032
6.434.2.85setProducerId	2032
6.434.2.86setReadOnlyBody	2032
6.434.2.87setReadOnlyProperties	2033
6.434.2.88setRecievedByDFBridge	2033
6.434.2.89setRedeliveryCounter	2033

6.434.2.90	setReplyTo	2033
6.434.2.91	setTargetConsumerId	2033
6.434.2.92	setTimestamp	2033
6.434.2.93	setTransactionId	2033
6.434.2.94	setType	2033
6.434.2.95	setUserID	2033
6.434.2.96	toString	2033
6.434.2.97	visit	2034
6.434.3	Field Documentation	2035
6.434.3.1	arrival	2035
6.434.3.2	brokerInTime	2035
6.434.3.3	brokerOutTime	2035
6.434.3.4	brokerPath	2035
6.434.3.5	cluster	2035
6.434.3.6	compressed	2035
6.434.3.7	content	2035
6.434.3.8	correlationId	2035
6.434.3.9	dataStructure	2035
6.434.3.10	DEFAULT_MESSAGE_SIZE	2035
6.434.3.11	destination	2035
6.434.3.12	droppable	2035
6.434.3.13	expiration	2035
6.434.3.14	groupId	2035
6.434.3.15	groupSequence	2035
6.434.3.16	ID_MESSAGE	2035
6.434.3.17	marshalledProperties	2035
6.434.3.18	messageId	2035
6.434.3.19	originalDestination	2035
6.434.3.20	originalTransactionId	2035
6.434.3.21	persistent	2035
6.434.3.22	priority	2035
6.434.3.23	producerId	2035
6.434.3.24	recievedByDFBridge	2035
6.434.3.25	redeliveryCounter	2035
6.434.3.26	replyTo	2035
6.434.3.27	targetConsumerId	2035

6.434.3.28	timestamp	2035
6.434.3.29	ransactionId	2035
6.434.3.30	type	2035
6.434.3.31	userId	2035
6.435	cms::Message Class Reference	2036
6.435.1	Detailed Description	2039
6.435.2	Constructor & Destructor Documentation	2040
6.435.2.1	~Message	2040
6.435.3	Member Function Documentation	2040
6.435.3.1	acknowledge	2040
6.435.3.2	clearBody	2040
6.435.3.3	clearProperties	2041
6.435.3.4	clone	2041
6.435.3.5	getBooleanProperty	2041
6.435.3.6	getByteProperty	2042
6.435.3.7	getCMSCorrelationID	2042
6.435.3.8	getCMSDeliveryMode	2042
6.435.3.9	getCMSDestination	2043
6.435.3.10	getCMSExpiration	2043
6.435.3.11	getCMSMessageID	2043
6.435.3.12	getCMSPriority	2044
6.435.3.13	getCMSRedelivered	2044
6.435.3.14	getCMSReplyTo	2045
6.435.3.15	getCMSTimestamp	2045
6.435.3.16	getCMSType	2045
6.435.3.17	getDoubleProperty	2046
6.435.3.18	getFloatProperty	2046
6.435.3.19	getIntProperty	2046
6.435.3.20	getLongProperty	2047
6.435.3.21	getPropertyNames	2047
6.435.3.22	getShortProperty	2047
6.435.3.23	getStringProperty	2048
6.435.3.24	propertyExists	2048
6.435.3.25	setBooleanProperty	2048
6.435.3.26	setByteProperty	2049
6.435.3.27	setCMSCorrelationID	2049

6.435.3.28	setCMSDeliveryMode	2050
6.435.3.29	setCMSDestination	2050
6.435.3.30	setCMSExpiration	2050
6.435.3.31	setCMSMessageID	2050
6.435.3.32	setCMSPriority	2051
6.435.3.33	setCMSRedelivered	2051
6.435.3.34	setCMSReplyTo	2051
6.435.3.35	setCMSTimestamp	2052
6.435.3.36	setCMSType	2052
6.435.3.37	setDoubleProperty	2053
6.435.3.38	setFloatProperty	2053
6.435.3.39	setIntProperty	2053
6.435.3.40	setLongProperty	2054
6.435.3.41	setShortProperty	2054
6.435.3.42	setStringProperty	2054
6.436	activemq::commands::MessageAck Class Reference	2055
6.436.1	Constructor & Destructor Documentation	2056
6.436.1.1	MessageAck	2056
6.436.1.2	MessageAck	2056
6.436.1.3	~MessageAck	2056
6.436.2	Member Function Documentation	2056
6.436.2.1	cloneDataStructure	2056
6.436.2.2	copyDataStructure	2057
6.436.2.3	equals	2057
6.436.2.4	getAckType	2057
6.436.2.5	getConsumerId	2057
6.436.2.6	getConsumerId	2057
6.436.2.7	getDataStructureType	2057
6.436.2.8	getDestination	2058
6.436.2.9	getDestination	2058
6.436.2.10	getFirstMessageId	2058
6.436.2.11	getFirstMessageId	2058
6.436.2.12	getLastMessageId	2058
6.436.2.13	getLastMessageId	2058
6.436.2.14	getMessageCount	2058
6.436.2.15	getTransactionId	2058

6.436.2.16	getTransactionId	2058
6.436.2.17	isMessageAck	2058
6.436.2.18	operator=	2059
6.436.2.19	setAckType	2059
6.436.2.20	setConsumerId	2059
6.436.2.21	setDestination	2059
6.436.2.22	setFirstMessageId	2059
6.436.2.23	setLastMessageId	2059
6.436.2.24	setMessageCount	2059
6.436.2.25	setTransactionId	2059
6.436.2.26	toString	2059
6.436.2.27	visit	2059
6.436.3	Field Documentation	2060
6.436.3.1	ackType	2060
6.436.3.2	consumerId	2060
6.436.3.3	destination	2060
6.436.3.4	firstMessageId	2060
6.436.3.5	ID_MESSAGEACK	2060
6.436.3.6	lastMessageId	2060
6.436.3.7	messageCount	2060
6.436.3.8	transactionId	2060
6.437	activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller Class Reference	2060
6.437.1	Detailed Description	2061
6.437.2	Constructor & Destructor Documentation	2061
6.437.2.1	MessageAckMarshaller	2061
6.437.2.2	~MessageAckMarshaller	2061
6.437.3	Member Function Documentation	2061
6.437.3.1	createObject	2061
6.437.3.2	getDataStructureType	2062
6.437.3.3	looseMarshal	2062
6.437.3.4	looseUnmarshal	2062
6.437.3.5	tightMarshal1	2063
6.437.3.6	tightMarshal2	2063
6.437.3.7	tightUnmarshal	2063
6.438	activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller Class Reference	2064
6.438.1	Detailed Description	2065

6.438.2 Constructor & Destructor Documentation	2065
6.438.2.1 MessageAckMarshaller	2065
6.438.2.2 ~MessageAckMarshaller	2065
6.438.3 Member Function Documentation	2065
6.438.3.1 createObject	2065
6.438.3.2 getDataStructureType	2065
6.438.3.3 looseMarshal	2066
6.438.3.4 looseUnmarshal	2066
6.438.3.5 tightMarshal1	2066
6.438.3.6 tightMarshal2	2067
6.438.3.7 tightUnmarshal	2067
6.439activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller Class Reference	2068
6.439.1 Detailed Description	2069
6.439.2 Constructor & Destructor Documentation	2069
6.439.2.1 MessageAckMarshaller	2069
6.439.2.2 ~MessageAckMarshaller	2069
6.439.3 Member Function Documentation	2069
6.439.3.1 createObject	2069
6.439.3.2 getDataStructureType	2069
6.439.3.3 looseMarshal	2070
6.439.3.4 looseUnmarshal	2070
6.439.3.5 tightMarshal1	2070
6.439.3.6 tightMarshal2	2071
6.439.3.7 tightUnmarshal	2071
6.440activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller Class Reference	2072
6.440.1 Detailed Description	2073
6.440.2 Constructor & Destructor Documentation	2073
6.440.2.1 MessageAckMarshaller	2073
6.440.2.2 ~MessageAckMarshaller	2073
6.440.3 Member Function Documentation	2073
6.440.3.1 createObject	2073
6.440.3.2 getDataStructureType	2073
6.440.3.3 looseMarshal	2074
6.440.3.4 looseUnmarshal	2074
6.440.3.5 tightMarshal1	2074
6.440.3.6 tightMarshal2	2075

6.440.3.7 tightUnmarshal	2075
6.441activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller Class Reference	2076
6.441.1 Detailed Description	2077
6.441.2 Constructor & Destructor Documentation	2077
6.441.2.1 MessageAckMarshaller	2077
6.441.2.2 ~MessageAckMarshaller	2077
6.441.3 Member Function Documentation	2077
6.441.3.1 createObject	2077
6.441.3.2 getDataStructureType	2077
6.441.3.3 looseMarshal	2078
6.441.3.4 looseUnmarshal	2078
6.441.3.5 tightMarshal1	2078
6.441.3.6 tightMarshal2	2079
6.441.3.7 tightUnmarshal	2079
6.442cms::MessageConsumer Class Reference	2080
6.442.1 Detailed Description	2080
6.442.2 Constructor & Destructor Documentation	2081
6.442.2.1 ~MessageConsumer	2081
6.442.3 Member Function Documentation	2081
6.442.3.1 getMessageListener	2081
6.442.3.2 getMessageSelector	2081
6.442.3.3 receive	2082
6.442.3.4 receive	2082
6.442.3.5 receiveNoWait	2082
6.442.3.6 setMessageListener	2082
6.443activemq::cmsutil::MessageCreator Class Reference	2083
6.443.1 Detailed Description	2083
6.443.2 Constructor & Destructor Documentation	2083
6.443.2.1 ~MessageCreator	2083
6.443.3 Member Function Documentation	2083
6.443.3.1 createMessage	2083
6.444activemq::commands::MessageDispatch Class Reference	2084
6.444.1 Constructor & Destructor Documentation	2085
6.444.1.1 MessageDispatch	2085
6.444.1.2 MessageDispatch	2085
6.444.1.3 ~MessageDispatch	2085

6.444.2 Member Function Documentation	2085
6.444.2.1 cloneDataStructure	2085
6.444.2.2 copyDataStructure	2085
6.444.2.3 equals	2086
6.444.2.4 getConsumerId	2086
6.444.2.5 getConsumerId	2086
6.444.2.6 getDataStructureType	2086
6.444.2.7 getDestination	2087
6.444.2.8 getDestination	2087
6.444.2.9 getMessage	2087
6.444.2.10getMessage	2087
6.444.2.11getRedeliveryCounter	2087
6.444.2.12sMessageDispatch	2087
6.444.2.13operator=	2087
6.444.2.14setConsumerId	2087
6.444.2.15setDestination	2087
6.444.2.16setMessage	2087
6.444.2.17setRedeliveryCounter	2087
6.444.2.18oString	2087
6.444.2.19visit	2088
6.444.3 Field Documentation	2088
6.444.3.1 consumerId	2088
6.444.3.2 destination	2088
6.444.3.3 ID_MESSAGEDISPATCH	2088
6.444.3.4 message	2088
6.444.3.5 redeliveryCounter	2088
6.445activemq::core::MessageDispatchChannel Class Reference	2088
6.445.1 Constructor & Destructor Documentation	2090
6.445.1.1 MessageDispatchChannel	2090
6.445.1.2 ~MessageDispatchChannel	2090
6.445.2 Member Function Documentation	2090
6.445.2.1 clear	2090
6.445.2.2 close	2090
6.445.2.3 dequeue	2090
6.445.2.4 dequeueNoWait	2091
6.445.2.5 enqueue	2091

6.445.2.6 enqueueFirst	2091
6.445.2.7 isClosed	2091
6.445.2.8 isEmpty	2091
6.445.2.9 isRunning	2092
6.445.2.10 lock	2092
6.445.2.11 notify	2092
6.445.2.12 notifyAll	2092
6.445.2.13 peek	2093
6.445.2.14 removeAll	2093
6.445.2.15 size	2093
6.445.2.16 start	2093
6.445.2.17 stop	2093
6.445.2.18 tryLock	2093
6.445.2.19 unlock	2094
6.445.2.20 wait	2094
6.445.2.21 wait	2094
6.445.2.22 wait	2095
6.446activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller Class	
Reference	2095
6.446.1 Detailed Description	2096
6.446.2 Constructor & Destructor Documentation	2097
6.446.2.1 MessageDispatchMarshaller	2097
6.446.2.2 ~MessageDispatchMarshaller	2097
6.446.3 Member Function Documentation	2097
6.446.3.1 createObject	2097
6.446.3.2 getDataStructureType	2097
6.446.3.3 looseMarshal	2097
6.446.3.4 looseUnmarshal	2098
6.446.3.5 tightMarshal1	2098
6.446.3.6 tightMarshal2	2098
6.446.3.7 tightUnmarshal	2099
6.447activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller Class	
Reference	2099
6.447.1 Detailed Description	2100
6.447.2 Constructor & Destructor Documentation	2101
6.447.2.1 MessageDispatchMarshaller	2101
6.447.2.2 ~MessageDispatchMarshaller	2101

6.447.3 Member Function Documentation	2101
6.447.3.1 createObject	2101
6.447.3.2 getDataStructureType	2101
6.447.3.3 looseMarshal	2101
6.447.3.4 looseUnmarshal	2102
6.447.3.5 tightMarshal1	2102
6.447.3.6 tightMarshal2	2102
6.447.3.7 tightUnmarshal	2103
6.448activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller Class	
Reference	2103
6.448.1 Detailed Description	2104
6.448.2 Constructor & Destructor Documentation	2105
6.448.2.1 MessageDispatchMarshaller	2105
6.448.2.2 ~MessageDispatchMarshaller	2105
6.448.3 Member Function Documentation	2105
6.448.3.1 createObject	2105
6.448.3.2 getDataStructureType	2105
6.448.3.3 looseMarshal	2105
6.448.3.4 looseUnmarshal	2106
6.448.3.5 tightMarshal1	2106
6.448.3.6 tightMarshal2	2106
6.448.3.7 tightUnmarshal	2107
6.449activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller Class	
Reference	2107
6.449.1 Detailed Description	2108
6.449.2 Constructor & Destructor Documentation	2109
6.449.2.1 MessageDispatchMarshaller	2109
6.449.2.2 ~MessageDispatchMarshaller	2109
6.449.3 Member Function Documentation	2109
6.449.3.1 createObject	2109
6.449.3.2 getDataStructureType	2109
6.449.3.3 looseMarshal	2109
6.449.3.4 looseUnmarshal	2110
6.449.3.5 tightMarshal1	2110
6.449.3.6 tightMarshal2	2110
6.449.3.7 tightUnmarshal	2111

6.450	activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller	Class	
	Reference		2111
6.450.1	Detailed Description		2112
6.450.2	Constructor & Destructor Documentation		2113
6.450.2.1	MessageDispatchMarshaller		2113
6.450.2.2	~MessageDispatchMarshaller		2113
6.450.3	Member Function Documentation		2113
6.450.3.1	createObject		2113
6.450.3.2	getDataStructureType		2113
6.450.3.3	looseMarshal		2113
6.450.3.4	looseUnmarshal		2114
6.450.3.5	tightMarshal1		2114
6.450.3.6	tightMarshal2		2114
6.450.3.7	tightUnmarshal		2115
6.451	activemq::commands::MessageDispatchNotification	Class Reference	2115
6.451.1	Constructor & Destructor Documentation		2117
6.451.1.1	MessageDispatchNotification		2117
6.451.1.2	MessageDispatchNotification		2117
6.451.1.3	~MessageDispatchNotification		2117
6.451.2	Member Function Documentation		2117
6.451.2.1	cloneDataStructure		2117
6.451.2.2	copyDataStructure		2117
6.451.2.3	equals		2117
6.451.2.4	getConsumerId		2118
6.451.2.5	getConsumerId		2118
6.451.2.6	getDataStructureType		2118
6.451.2.7	getDeliverySequenceId		2118
6.451.2.8	getDestination		2118
6.451.2.9	getDestination		2118
6.451.2.10	getMessageId		2118
6.451.2.11	getMessageId		2118
6.451.2.12	setMessageDispatchNotification		2118
6.451.2.13	operator=		2119
6.451.2.14	setConsumerId		2119
6.451.2.15	setDeliverySequenceId		2119
6.451.2.16	setDestination		2119
6.451.2.17	setMessageId		2119

6.451.2.18	oString	2119
6.451.2.19	visit	2119
6.451.3	Field Documentation	2120
6.451.3.1	consumerId	2120
6.451.3.2	deliverySequenceId	2120
6.451.3.3	destination	2120
6.451.3.4	ID_ MESSAGEDISPATCHNOTIFICATION	2120
6.451.3.5	messageId	2120
6.452	activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller	
	Class Reference	2120
6.452.1	Detailed Description	2121
6.452.2	Constructor & Destructor Documentation	2121
6.452.2.1	MessageDispatchNotificationMarshaller	2121
6.452.2.2	~MessageDispatchNotificationMarshaller	2121
6.452.3	Member Function Documentation	2121
6.452.3.1	createObject	2121
6.452.3.2	getDataStructureType	2122
6.452.3.3	looseMarshal	2122
6.452.3.4	looseUnmarshal	2122
6.452.3.5	tightMarshal1	2123
6.452.3.6	tightMarshal2	2123
6.452.3.7	tightUnmarshal	2123
6.453	activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller	
	Class Reference	2124
6.453.1	Detailed Description	2125
6.453.2	Constructor & Destructor Documentation	2125
6.453.2.1	MessageDispatchNotificationMarshaller	2125
6.453.2.2	~MessageDispatchNotificationMarshaller	2125
6.453.3	Member Function Documentation	2125
6.453.3.1	createObject	2125
6.453.3.2	getDataStructureType	2125
6.453.3.3	looseMarshal	2126
6.453.3.4	looseUnmarshal	2126
6.453.3.5	tightMarshal1	2126
6.453.3.6	tightMarshal2	2127
6.453.3.7	tightUnmarshal	2127

6.454	activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller	
	Class Reference	2128
6.454.1	Detailed Description	2129
6.454.2	Constructor & Destructor Documentation	2129
	6.454.2.1 MessageDispatchNotificationMarshaller	2129
	6.454.2.2 ~MessageDispatchNotificationMarshaller	2129
6.454.3	Member Function Documentation	2129
	6.454.3.1 createObject	2129
	6.454.3.2 getDataStructureType	2129
	6.454.3.3 looseMarshal	2130
	6.454.3.4 looseUnmarshal	2130
	6.454.3.5 tightMarshal1	2130
	6.454.3.6 tightMarshal2	2131
	6.454.3.7 tightUnmarshal	2131
6.455	activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller	
	Class Reference	2132
6.455.1	Detailed Description	2133
6.455.2	Constructor & Destructor Documentation	2133
	6.455.2.1 MessageDispatchNotificationMarshaller	2133
	6.455.2.2 ~MessageDispatchNotificationMarshaller	2133
6.455.3	Member Function Documentation	2133
	6.455.3.1 createObject	2133
	6.455.3.2 getDataStructureType	2133
	6.455.3.3 looseMarshal	2134
	6.455.3.4 looseUnmarshal	2134
	6.455.3.5 tightMarshal1	2134
	6.455.3.6 tightMarshal2	2135
	6.455.3.7 tightUnmarshal	2135
6.456	activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller	
	Class Reference	2136
6.456.1	Detailed Description	2137
6.456.2	Constructor & Destructor Documentation	2137
	6.456.2.1 MessageDispatchNotificationMarshaller	2137
	6.456.2.2 ~MessageDispatchNotificationMarshaller	2137
6.456.3	Member Function Documentation	2137
	6.456.3.1 createObject	2137
	6.456.3.2 getDataStructureType	2137

6.456.3.3 looseMarshal	2138
6.456.3.4 looseUnmarshal	2138
6.456.3.5 tightMarshal1	2138
6.456.3.6 tightMarshal2	2139
6.456.3.7 tightUnmarshal	2139
6.457cms::MessageEOFException Class Reference	2140
6.457.1 Detailed Description	2140
6.457.2 Constructor & Destructor Documentation	2141
6.457.2.1 MessageEOFException	2141
6.457.2.2 MessageEOFException	2141
6.457.2.3 MessageEOFException	2141
6.457.2.4 MessageEOFException	2141
6.457.2.5 ~MessageEOFException	2141
6.458cms::MessageFormatException Class Reference	2141
6.458.1 Detailed Description	2141
6.458.2 Constructor & Destructor Documentation	2142
6.458.2.1 MessageFormatException	2142
6.458.2.2 MessageFormatException	2142
6.458.2.3 MessageFormatException	2142
6.458.2.4 MessageFormatException	2142
6.458.2.5 ~MessageFormatException	2142
6.459activemq::commands::MessageId Class Reference	2142
6.459.1 Member Typedef Documentation	2143
6.459.1.1 COMPARATOR	2143
6.459.2 Constructor & Destructor Documentation	2143
6.459.2.1 MessageId	2143
6.459.2.2 MessageId	2143
6.459.2.3 ~MessageId	2143
6.459.3 Member Function Documentation	2143
6.459.3.1 cloneDataStructure	2143
6.459.3.2 compareTo	2144
6.459.3.3 copyDataStructure	2144
6.459.3.4 equals	2144
6.459.3.5 equals	2144
6.459.3.6 getBrokerSequenceId	2144
6.459.3.7 getDataStructureType	2144

6.459.3.8	getProducerId	2145
6.459.3.9	getProducerId	2145
6.459.3.10	getProducerSequenceId	2145
6.459.3.11	operator<	2145
6.459.3.12	operator=	2145
6.459.3.13	operator==	2145
6.459.3.14	setBrokerSequenceId	2145
6.459.3.15	setProducerId	2145
6.459.3.16	setProducerSequenceId	2145
6.459.3.17	toString	2145
6.459.4	Field Documentation	2146
6.459.4.1	brokerSequenceId	2146
6.459.4.2	ID_MESSAGEID	2146
6.459.4.3	producerId	2146
6.459.4.4	producerSequenceId	2146
6.460	activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller Class Reference	2146
6.460.1	Detailed Description	2147
6.460.2	Constructor & Destructor Documentation	2147
6.460.2.1	MessageIdMarshaller	2147
6.460.2.2	~MessageIdMarshaller	2147
6.460.3	Member Function Documentation	2147
6.460.3.1	createObject	2147
6.460.3.2	getDataStructureType	2147
6.460.3.3	looseMarshal	2148
6.460.3.4	looseUnmarshal	2148
6.460.3.5	tightMarshal1	2148
6.460.3.6	tightMarshal2	2149
6.460.3.7	tightUnmarshal	2149
6.461	activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller Class Reference	2150
6.461.1	Detailed Description	2151
6.461.2	Constructor & Destructor Documentation	2151
6.461.2.1	MessageIdMarshaller	2151
6.461.2.2	~MessageIdMarshaller	2151
6.461.3	Member Function Documentation	2151
6.461.3.1	createObject	2151
6.461.3.2	getDataStructureType	2151

6.461.3.3 looseMarshal	2152
6.461.3.4 looseUnmarshal	2152
6.461.3.5 tightMarshal1	2152
6.461.3.6 tightMarshal2	2153
6.461.3.7 tightUnmarshal	2153
6.462activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller Class Reference	2154
6.462.1 Detailed Description	2155
6.462.2 Constructor & Destructor Documentation	2155
6.462.2.1 MessageIdMarshaller	2155
6.462.2.2 ~MessageIdMarshaller	2155
6.462.3 Member Function Documentation	2155
6.462.3.1 createObject	2155
6.462.3.2 getDataStructureType	2155
6.462.3.3 looseMarshal	2155
6.462.3.4 looseUnmarshal	2156
6.462.3.5 tightMarshal1	2156
6.462.3.6 tightMarshal2	2157
6.462.3.7 tightUnmarshal	2157
6.463activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller Class Reference	2157
6.463.1 Detailed Description	2158
6.463.2 Constructor & Destructor Documentation	2159
6.463.2.1 MessageIdMarshaller	2159
6.463.2.2 ~MessageIdMarshaller	2159
6.463.3 Member Function Documentation	2159
6.463.3.1 createObject	2159
6.463.3.2 getDataStructureType	2159
6.463.3.3 looseMarshal	2159
6.463.3.4 looseUnmarshal	2160
6.463.3.5 tightMarshal1	2160
6.463.3.6 tightMarshal2	2160
6.463.3.7 tightUnmarshal	2161
6.464activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller Class Reference	2161
6.464.1 Detailed Description	2162
6.464.2 Constructor & Destructor Documentation	2163
6.464.2.1 MessageIdMarshaller	2163
6.464.2.2 ~MessageIdMarshaller	2163

6.464.3 Member Function Documentation	2163
6.464.3.1 createObject	2163
6.464.3.2 getDataStructureType	2163
6.464.3.3 looseMarshal	2163
6.464.3.4 looseUnmarshal	2164
6.464.3.5 tightMarshal1	2164
6.464.3.6 tightMarshal2	2164
6.464.3.7 tightUnmarshal	2165
6.465 cms::MessageListener Class Reference	2165
6.465.1 Detailed Description	2166
6.465.2 Constructor & Destructor Documentation	2166
6.465.2.1 ~MessageListener	2166
6.465.3 Member Function Documentation	2166
6.465.3.1 onMessage	2166
6.466 activemq::wireformat::openwire::marshal::v2::MessageMarshaller Class Reference	2166
6.466.1 Detailed Description	2167
6.466.2 Constructor & Destructor Documentation	2167
6.466.2.1 MessageMarshaller	2167
6.466.2.2 ~MessageMarshaller	2167
6.466.3 Member Function Documentation	2167
6.466.3.1 looseMarshal	2167
6.466.3.2 looseUnmarshal	2168
6.466.3.3 tightMarshal1	2169
6.466.3.4 tightMarshal2	2169
6.466.3.5 tightUnmarshal	2170
6.467 activemq::wireformat::openwire::marshal::v4::MessageMarshaller Class Reference	2170
6.467.1 Detailed Description	2171
6.467.2 Constructor & Destructor Documentation	2172
6.467.2.1 MessageMarshaller	2172
6.467.2.2 ~MessageMarshaller	2172
6.467.3 Member Function Documentation	2172
6.467.3.1 looseMarshal	2172
6.467.3.2 looseUnmarshal	2172
6.467.3.3 tightMarshal1	2173
6.467.3.4 tightMarshal2	2174
6.467.3.5 tightUnmarshal	2174

6.468	activemq::wireformat::openwire::marshal::v3::MessageMarshaller Class Reference	2175
6.468.1	Detailed Description	2176
6.468.2	Constructor & Destructor Documentation	2176
6.468.2.1	MessageMarshaller	2176
6.468.2.2	~MessageMarshaller	2176
6.468.3	Member Function Documentation	2176
6.468.3.1	looseMarshal	2176
6.468.3.2	looseUnmarshal	2176
6.468.3.3	tightMarshal1	2177
6.468.3.4	tightMarshal2	2178
6.468.3.5	tightUnmarshal	2178
6.469	activemq::wireformat::openwire::marshal::v5::MessageMarshaller Class Reference	2179
6.469.1	Detailed Description	2180
6.469.2	Constructor & Destructor Documentation	2180
6.469.2.1	MessageMarshaller	2180
6.469.2.2	~MessageMarshaller	2180
6.469.3	Member Function Documentation	2180
6.469.3.1	looseMarshal	2180
6.469.3.2	looseUnmarshal	2180
6.469.3.3	tightMarshal1	2181
6.469.3.4	tightMarshal2	2182
6.469.3.5	tightUnmarshal	2182
6.470	activemq::wireformat::openwire::marshal::v1::MessageMarshaller Class Reference	2183
6.470.1	Detailed Description	2184
6.470.2	Constructor & Destructor Documentation	2184
6.470.2.1	MessageMarshaller	2184
6.470.2.2	~MessageMarshaller	2184
6.470.3	Member Function Documentation	2184
6.470.3.1	looseMarshal	2184
6.470.3.2	looseUnmarshal	2184
6.470.3.3	tightMarshal1	2185
6.470.3.4	tightMarshal2	2186
6.470.3.5	tightUnmarshal	2186
6.471	cms::MessageNotReadableException Class Reference	2187
6.471.1	Detailed Description	2187
6.471.2	Constructor & Destructor Documentation	2188

6.471.2.1 MessageNotReadableException	2188
6.471.2.2 MessageNotReadableException	2188
6.471.2.3 MessageNotReadableException	2188
6.471.2.4 MessageNotReadableException	2188
6.471.2.5 ~MessageNotReadableException	2188
6.472cms::MessageNotWriteableException Class Reference	2188
6.472.1 Detailed Description	2188
6.472.2 Constructor & Destructor Documentation	2189
6.472.2.1 MessageNotWriteableException	2189
6.472.2.2 MessageNotWriteableException	2189
6.472.2.3 MessageNotWriteableException	2189
6.472.2.4 MessageNotWriteableException	2189
6.472.2.5 ~MessageNotWriteableException	2189
6.473cms::MessageProducer Class Reference	2189
6.473.1 Detailed Description	2190
6.473.2 Constructor & Destructor Documentation	2191
6.473.2.1 ~MessageProducer	2191
6.473.3 Member Function Documentation	2191
6.473.3.1 getDeliveryMode	2191
6.473.3.2 getDisableMessageID	2191
6.473.3.3 getDisableMessageTimeStamp	2192
6.473.3.4 getPriority	2192
6.473.3.5 getTimeToLive	2192
6.473.3.6 send	2192
6.473.3.7 send	2193
6.473.3.8 send	2194
6.473.3.9 send	2194
6.473.3.10 setDeliveryMode	2195
6.473.3.11 setDisableMessageID	2195
6.473.3.12 setDisableMessageTimeStamp	2195
6.473.3.13 setPriority	2195
6.473.3.14 setTimeToLive	2196
6.474activemq::wireformat::openwire::utils::MessagePropertyInterceptor Class Reference	2196
6.474.1 Detailed Description	2198
6.474.2 Constructor & Destructor Documentation	2198
6.474.2.1 MessagePropertyInterceptor	2198

6.474.2.2	~MessagePropertyInterceptor	2198
6.474.3	Member Function Documentation	2198
6.474.3.1	getBooleanProperty	2198
6.474.3.2	getByteProperty	2198
6.474.3.3	getDoubleProperty	2199
6.474.3.4	getFloatProperty	2199
6.474.3.5	getIntProperty	2199
6.474.3.6	getLongProperty	2200
6.474.3.7	getShortProperty	2200
6.474.3.8	getStringProperty	2200
6.474.3.9	setBooleanProperty	2200
6.474.3.10	setByteProperty	2201
6.474.3.11	setDoubleProperty	2201
6.474.3.12	setFloatProperty	2201
6.474.3.13	setIntProperty	2201
6.474.3.14	setLongProperty	2201
6.474.3.15	setShortProperty	2202
6.474.3.16	setStringProperty	2202
6.475	activemq::commands::MessagePull Class Reference	2202
6.475.1	Constructor & Destructor Documentation	2204
6.475.1.1	MessagePull	2204
6.475.1.2	MessagePull	2204
6.475.1.3	~MessagePull	2204
6.475.2	Member Function Documentation	2204
6.475.2.1	cloneDataStructure	2204
6.475.2.2	copyDataStructure	2204
6.475.2.3	equals	2204
6.475.2.4	getConsumerId	2205
6.475.2.5	getConsumerId	2205
6.475.2.6	getCorrelationId	2205
6.475.2.7	getCorrelationId	2205
6.475.2.8	getDataStructureType	2205
6.475.2.9	getDestination	2206
6.475.2.10	getDestination	2206
6.475.2.11	getMessageId	2206
6.475.2.12	getMessageId	2206

6.475.2.13	getTimeout	2206
6.475.2.14	operator=	2206
6.475.2.15	setConsumerId	2206
6.475.2.16	setCorrelationId	2206
6.475.2.17	setDestination	2206
6.475.2.18	setMessageId	2206
6.475.2.19	setTimeout	2206
6.475.2.20	toString	2206
6.475.2.21	visit	2206
6.475.3	Field Documentation	2207
6.475.3.1	consumerId	2207
6.475.3.2	correlationId	2207
6.475.3.3	destination	2207
6.475.3.4	ID_MESSAGEPULL	2207
6.475.3.5	messageId	2207
6.475.3.6	timeout	2207
6.476	activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller Class Reference	2207
6.476.1	Detailed Description	2208
6.476.2	Constructor & Destructor Documentation	2208
6.476.2.1	MessagePullMarshaller	2208
6.476.2.2	~MessagePullMarshaller	2208
6.476.3	Member Function Documentation	2208
6.476.3.1	createObject	2208
6.476.3.2	getDataStructureType	2209
6.476.3.3	looseMarshal	2209
6.476.3.4	looseUnmarshal	2209
6.476.3.5	tightMarshal1	2210
6.476.3.6	tightMarshal2	2210
6.476.3.7	tightUnmarshal	2210
6.477	activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller Class Reference	2211
6.477.1	Detailed Description	2212
6.477.2	Constructor & Destructor Documentation	2212
6.477.2.1	MessagePullMarshaller	2212
6.477.2.2	~MessagePullMarshaller	2212
6.477.3	Member Function Documentation	2212
6.477.3.1	createObject	2212

6.477.3.2	getDataStructureType	2212
6.477.3.3	looseMarshal	2213
6.477.3.4	looseUnmarshal	2213
6.477.3.5	tightMarshal1	2213
6.477.3.6	tightMarshal2	2214
6.477.3.7	tightUnmarshal	2214
6.478	activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller Class Reference	2215
6.478.1	Detailed Description	2216
6.478.2	Constructor & Destructor Documentation	2216
6.478.2.1	MessagePullMarshaller	2216
6.478.2.2	~MessagePullMarshaller	2216
6.478.3	Member Function Documentation	2216
6.478.3.1	createObject	2216
6.478.3.2	getDataStructureType	2216
6.478.3.3	looseMarshal	2217
6.478.3.4	looseUnmarshal	2217
6.478.3.5	tightMarshal1	2217
6.478.3.6	tightMarshal2	2218
6.478.3.7	tightUnmarshal	2218
6.479	activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller Class Reference	2219
6.479.1	Detailed Description	2220
6.479.2	Constructor & Destructor Documentation	2220
6.479.2.1	MessagePullMarshaller	2220
6.479.2.2	~MessagePullMarshaller	2220
6.479.3	Member Function Documentation	2220
6.479.3.1	createObject	2220
6.479.3.2	getDataStructureType	2220
6.479.3.3	looseMarshal	2221
6.479.3.4	looseUnmarshal	2221
6.479.3.5	tightMarshal1	2221
6.479.3.6	tightMarshal2	2222
6.479.3.7	tightUnmarshal	2222
6.480	activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller Class Reference	2223
6.480.1	Detailed Description	2224
6.480.2	Constructor & Destructor Documentation	2224
6.480.2.1	MessagePullMarshaller	2224

6.480.2.2 ~MessagePullMarshaller	2224
6.480.3 Member Function Documentation	2224
6.480.3.1 createObject	2224
6.480.3.2 getDataStructureType	2224
6.480.3.3 looseMarshal	2225
6.480.3.4 looseUnmarshal	2225
6.480.3.5 tightMarshal1	2225
6.480.3.6 tightMarshal2	2226
6.480.3.7 tightUnmarshal	2226
6.481activemq::transport::mock::MockTransport Class Reference	2227
6.481.1 Detailed Description	2229
6.481.2 Constructor & Destructor Documentation	2229
6.481.2.1 MockTransport	2229
6.481.2.2 ~MockTransport	2229
6.481.3 Member Function Documentation	2229
6.481.3.1 close	2229
6.481.3.2 fireCommand	2230
6.481.3.3 fireException	2230
6.481.3.4 getInstance	2231
6.481.3.5 getNumReceivedMessageBeforeFail	2231
6.481.3.6 getNumReceivedMessages	2231
6.481.3.7 getNumSentKeepAlives	2231
6.481.3.8 getNumSentKeepAlivesBeforeFail	2231
6.481.3.9 getNumSentMessageBeforeFail	2231
6.481.3.10getNumSentMessages	2231
6.481.3.11getRemoteAddress	2231
6.481.3.12getTransportListener	2231
6.481.3.13getWireFormat	2231
6.481.3.14sClosed	2232
6.481.3.15sConnected	2232
6.481.3.16sFailOnClose	2232
6.481.3.17sFailOnKeepAliveSends	2232
6.481.3.18sFailOnReceiveMessage	2232
6.481.3.19sFailOnSendMessage	2232
6.481.3.20sFailOnStart	2232
6.481.3.21sFailOnStop	2232

6.481.3.22	isFaultTolerant	2232
6.481.3.23	narrow	2233
6.481.3.24	neway	2233
6.481.3.25	reconnect	2233
6.481.3.26	request	2234
6.481.3.27	request	2234
6.481.3.28	setFailOnClose	2235
6.481.3.29	setFailOnKeepAliveSends	2235
6.481.3.30	setFailOnReceiveMessage	2235
6.481.3.31	setFailOnSendMessage	2235
6.481.3.32	setFailOnStart	2235
6.481.3.33	setFailOnStop	2235
6.481.3.34	setNumReceivedMessageBeforeFail	2235
6.481.3.35	setNumReceivedMessages	2235
6.481.3.36	setNumSentKeepAlives	2235
6.481.3.37	setNumSentKeepAlivesBeforeFail	2235
6.481.3.38	setNumSentMessageBeforeFail	2235
6.481.3.39	setNumSentMessages	2235
6.481.3.40	setOutgoingListener	2235
6.481.3.41	setResponseBuilder	2236
6.481.3.42	setTransportListener	2236
6.481.3.43	setWireFormat	2236
6.481.3.44	start	2236
6.481.3.45	stop	2236
6.482	activemq::transport::mock::MockTransportFactory Class Reference	2237
6.482.1	Detailed Description	2237
6.482.2	Constructor & Destructor Documentation	2238
6.482.2.1	~MockTransportFactory	2238
6.482.3	Member Function Documentation	2238
6.482.3.1	create	2238
6.482.3.2	createComposite	2238
6.482.3.3	doCreateComposite	2238
6.483	decaf::util::concurrent::Mutex Class Reference	2239
6.483.1	Detailed Description	2240
6.483.2	Constructor & Destructor Documentation	2240
6.483.2.1	Mutex	2240

6.483.2.2 ~Mutex	2240
6.483.3 Member Function Documentation	2240
6.483.3.1 lock	2240
6.483.3.2 notify	2241
6.483.3.3 notifyAll	2241
6.483.3.4 tryLock	2241
6.483.3.5 unlock	2242
6.483.3.6 wait	2242
6.483.3.7 wait	2243
6.483.3.8 wait	2243
6.484decaf::util::concurrent::MutexHandle Class Reference	2244
6.484.1 Constructor & Destructor Documentation	2244
6.484.1.1 MutexHandle	2244
6.484.1.2 ~MutexHandle	2244
6.484.1.3 MutexHandle	2244
6.484.1.4 ~MutexHandle	2244
6.484.2 Field Documentation	2244
6.484.2.1 lock_count	2244
6.484.2.2 lock_owner	2244
6.484.2.3 mutex	2244
6.484.2.4 mutex	2244
6.485decaf::internal::util::concurrent::MutexImpl Class Reference	2244
6.485.1 Member Function Documentation	2245
6.485.1.1 create	2245
6.485.1.2 destroy	2245
6.485.1.3 lock	2245
6.485.1.4 trylock	2246
6.485.1.5 unlock	2246
6.486activemq::commands::NetworkBridgeFilter Class Reference	2246
6.486.1 Constructor & Destructor Documentation	2247
6.486.1.1 NetworkBridgeFilter	2247
6.486.1.2 NetworkBridgeFilter	2247
6.486.1.3 ~NetworkBridgeFilter	2247
6.486.2 Member Function Documentation	2247
6.486.2.1 cloneDataStructure	2247
6.486.2.2 copyDataStructure	2248

6.486.2.3 equals	2248
6.486.2.4 getDataStructureType	2248
6.486.2.5 getNetworkBrokerId	2249
6.486.2.6 getNetworkBrokerId	2249
6.486.2.7 getNetworkTTL	2249
6.486.2.8 operator=	2249
6.486.2.9 setNetworkBrokerId	2249
6.486.2.10 setNetworkTTL	2249
6.486.2.11 toString	2249
6.486.3 Field Documentation	2249
6.486.3.1 ID_NETWORKBRIDGEFILTER	2249
6.486.3.2 networkBrokerId	2249
6.486.3.3 networkTTL	2249
6.487activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller Class	
Reference	2250
6.487.1 Detailed Description	2251
6.487.2 Constructor & Destructor Documentation	2251
6.487.2.1 NetworkBridgeFilterMarshaller	2251
6.487.2.2 ~NetworkBridgeFilterMarshaller	2251
6.487.3 Member Function Documentation	2251
6.487.3.1 createObject	2251
6.487.3.2 getDataStructureType	2251
6.487.3.3 looseMarshal	2251
6.487.3.4 looseUnmarshal	2252
6.487.3.5 tightMarshal1	2252
6.487.3.6 tightMarshal2	2253
6.487.3.7 tightUnmarshal	2253
6.488activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller Class	
Reference	2253
6.488.1 Detailed Description	2254
6.488.2 Constructor & Destructor Documentation	2255
6.488.2.1 NetworkBridgeFilterMarshaller	2255
6.488.2.2 ~NetworkBridgeFilterMarshaller	2255
6.488.3 Member Function Documentation	2255
6.488.3.1 createObject	2255
6.488.3.2 getDataStructureType	2255
6.488.3.3 looseMarshal	2255

6.488.3.4 looseUnmarshal	2256
6.488.3.5 tightMarshal1	2256
6.488.3.6 tightMarshal2	2256
6.488.3.7 tightUnmarshal	2257
6.489activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller Class	
Reference	2257
6.489.1 Detailed Description	2258
6.489.2 Constructor & Destructor Documentation	2259
6.489.2.1 NetworkBridgeFilterMarshaller	2259
6.489.2.2 ~NetworkBridgeFilterMarshaller	2259
6.489.3 Member Function Documentation	2259
6.489.3.1 createObject	2259
6.489.3.2 getDataStructureType	2259
6.489.3.3 looseMarshal	2259
6.489.3.4 looseUnmarshal	2260
6.489.3.5 tightMarshal1	2260
6.489.3.6 tightMarshal2	2260
6.489.3.7 tightUnmarshal	2261
6.490activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller Class	
Reference	2261
6.490.1 Detailed Description	2262
6.490.2 Constructor & Destructor Documentation	2263
6.490.2.1 NetworkBridgeFilterMarshaller	2263
6.490.2.2 ~NetworkBridgeFilterMarshaller	2263
6.490.3 Member Function Documentation	2263
6.490.3.1 createObject	2263
6.490.3.2 getDataStructureType	2263
6.490.3.3 looseMarshal	2263
6.490.3.4 looseUnmarshal	2264
6.490.3.5 tightMarshal1	2264
6.490.3.6 tightMarshal2	2264
6.490.3.7 tightUnmarshal	2265
6.491activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller Class	
Reference	2265
6.491.1 Detailed Description	2266
6.491.2 Constructor & Destructor Documentation	2267
6.491.2.1 NetworkBridgeFilterMarshaller	2267

6.491.2.2 ~NetworkBridgeFilterMarshaller	2267
6.491.3 Member Function Documentation	2267
6.491.3.1 createObject	2267
6.491.3.2 getDataStructureType	2267
6.491.3.3 looseMarshal	2267
6.491.3.4 looseUnmarshal	2268
6.491.3.5 tightMarshal1	2268
6.491.3.6 tightMarshal2	2268
6.491.3.7 tightUnmarshal	2269
6.492decaf::net::NoRouteToHostException Class Reference	2269
6.492.1 Constructor & Destructor Documentation	2270
6.492.1.1 NoRouteToHostException	2270
6.492.1.2 NoRouteToHostException	2270
6.492.1.3 NoRouteToHostException	2270
6.492.1.4 NoRouteToHostException	2271
6.492.1.5 NoRouteToHostException	2271
6.492.1.6 NoRouteToHostException	2271
6.492.1.7 ~NoRouteToHostException	2271
6.492.2 Member Function Documentation	2271
6.492.2.1 clone	2271
6.493decaf::security::NoSuchAlgorithmException Class Reference	2272
6.493.1 Constructor & Destructor Documentation	2273
6.493.1.1 NoSuchAlgorithmException	2273
6.493.1.2 NoSuchAlgorithmException	2273
6.493.1.3 NoSuchAlgorithmException	2273
6.493.1.4 NoSuchAlgorithmException	2273
6.493.1.5 NoSuchAlgorithmException	2273
6.493.1.6 NoSuchAlgorithmException	2274
6.493.1.7 ~NoSuchAlgorithmException	2274
6.493.2 Member Function Documentation	2274
6.493.2.1 clone	2274
6.494decaf::lang::exceptions::NoSuchElementException Class Reference	2274
6.494.1 Constructor & Destructor Documentation	2275
6.494.1.1 NoSuchElementException	2275
6.494.1.2 NoSuchElementException	2275
6.494.1.3 NoSuchElementException	2275

6.494.1.4 NoSuchElementException	2276
6.494.1.5 NoSuchElementException	2276
6.494.1.6 NoSuchElementException	2276
6.494.1.7 ~NoSuchElementException	2276
6.494.2 Member Function Documentation	2276
6.494.2.1 clone	2276
6.495decaf::security::NoSuchProviderException Class Reference	2277
6.495.1 Constructor & Destructor Documentation	2278
6.495.1.1 NoSuchProviderException	2278
6.495.1.2 NoSuchProviderException	2278
6.495.1.3 NoSuchProviderException	2278
6.495.1.4 NoSuchProviderException	2278
6.495.1.5 NoSuchProviderException	2278
6.495.1.6 NoSuchProviderException	2279
6.495.1.7 ~NoSuchProviderException	2279
6.495.2 Member Function Documentation	2279
6.495.2.1 clone	2279
6.496decaf::lang::exceptions::NullPointerException Class Reference	2279
6.496.1 Constructor & Destructor Documentation	2280
6.496.1.1 NullPointerException	2280
6.496.1.2 NullPointerException	2280
6.496.1.3 NullPointerException	2280
6.496.1.4 NullPointerException	2281
6.496.1.5 NullPointerException	2281
6.496.1.6 NullPointerException	2281
6.496.1.7 ~NullPointerException	2281
6.496.2 Member Function Documentation	2281
6.496.2.1 clone	2281
6.497decaf::lang::Number Class Reference	2282
6.497.1 Detailed Description	2282
6.497.2 Constructor & Destructor Documentation	2283
6.497.2.1 ~Number	2283
6.497.3 Member Function Documentation	2283
6.497.3.1 byteValue	2283
6.497.3.2 doubleValue	2283
6.497.3.3 floatValue	2283

6.497.3.4 int Value	2283
6.497.3.5 long Value	2283
6.497.3.6 short Value	2284
6.498decaf::lang::exceptions::NumberFormatException Class Reference	2284
6.498.1 Constructor & Destructor Documentation	2285
6.498.1.1 NumberFormatException	2285
6.498.1.2 NumberFormatException	2285
6.498.1.3 NumberFormatException	2285
6.498.1.4 NumberFormatException	2285
6.498.1.5 NumberFormatException	2286
6.498.1.6 NumberFormatException	2286
6.498.1.7 ~NumberFormatException	2286
6.498.2 Member Function Documentation	2286
6.498.2.1 clone	2286
6.499cms::ObjectMessage Class Reference	2287
6.499.1 Detailed Description	2287
6.499.2 Constructor & Destructor Documentation	2287
6.499.2.1 ~ObjectMessage	2287
6.500decaf::security_provider::unix::openssl::OpenSSLX500Principal Class Reference	2287
6.500.1 Detailed Description	2288
6.500.2 Constructor & Destructor Documentation	2288
6.500.2.1 OpenSSLX500Principal	2288
6.500.2.2 ~OpenSSLX500Principal	2288
6.500.3 Member Function Documentation	2289
6.500.3.1 equals	2289
6.500.3.2 getEncoded	2289
6.500.3.3 getEncoded	2289
6.500.3.4 getName	2289
6.500.3.5 getX509Name	2290
6.500.3.6 toString	2290
6.501decaf::security_provider::unix::openssl::OpenSSLX509Certificate Class Reference	2290
6.501.1 Constructor & Destructor Documentation	2291
6.501.1.1 ~OpenSSLX509Certificate	2291
6.501.2 Member Function Documentation	2291
6.501.2.1 checkValidity	2291
6.501.2.2 checkValidity	2292

6.501.2.3 equals	2292
6.501.2.4 getBasicConstraints	2292
6.501.2.5 getEncoded	2292
6.501.2.6 getIssuerUniqueID	2292
6.501.2.7 getIssuerX500Principal	2293
6.501.2.8 getKeyUsage	2293
6.501.2.9 getNotAfter	2293
6.501.2.10 getNotBefore	2293
6.501.2.11 getPublicKey	2293
6.501.2.12 getPublicKey	2293
6.501.2.13 getSigAlgName	2294
6.501.2.14 getSigAlgOID	2294
6.501.2.15 getSigAlgParams	2294
6.501.2.16 getSignature	2294
6.501.2.17 getSubjectUniqueID	2294
6.501.2.18 getSubjectX500Principal	2294
6.501.2.19 getTBSCertificate	2294
6.501.2.20 getType	2294
6.501.2.21 getVersion	2295
6.501.2.22 toString	2295
6.501.2.23 verify	2295
6.501.2.24 verify	2296
6.502activemq::wireformat::openwire::OpenWireFormat Class Reference	2296
6.502.1 Constructor & Destructor Documentation	2299
6.502.1.1 OpenWireFormat	2299
6.502.1.2 ~OpenWireFormat	2300
6.502.2 Member Function Documentation	2300
6.502.2.1 addMarshaller	2300
6.502.2.2 createNegotiator	2300
6.502.2.3 destroyMarshalers	2300
6.502.2.4 doUnmarshal	2300
6.502.2.5 getCacheSize	2301
6.502.2.6 getMaxInactivityDuration	2301
6.502.2.7 getMaxInactivityDurationInitialDelay	2301
6.502.2.8 getPreferredWireFormatInfo	2301
6.502.2.9 getVersion	2302

6.502.2.10	hasNegotiator	2302
6.502.2.11	linReceive	2302
6.502.2.12	sCacheEnabled	2302
6.502.2.13	sSizePrefixDisabled	2302
6.502.2.14	sStackTraceEnabled	2303
6.502.2.15	sTcpNoDelayEnabled	2303
6.502.2.16	sTightEncodingEnabled	2303
6.502.2.17	ooseMarshalNestedObject	2303
6.502.2.18	ooseUnmarshalNestedObject	2304
6.502.2.19	marshal	2304
6.502.2.20	renegotiateWireFormat	2304
6.502.2.21	set CacheEnabled	2305
6.502.2.22	set CacheSize	2305
6.502.2.23	set MaxInactivityDuration	2305
6.502.2.24	set MaxInactivityDurationInitialDelay	2305
6.502.2.25	set PreferredWireFormatInfo	2305
6.502.2.26	set SizePrefixDisabled	2306
6.502.2.27	set StackTraceEnabled	2306
6.502.2.28	set TcpNoDelayEnabled	2306
6.502.2.29	set TightEncodingEnabled	2306
6.502.2.30	set Version	2306
6.502.2.31	tightMarshalNestedObject1	2307
6.502.2.32	tightMarshalNestedObject2	2307
6.502.2.33	tightUnmarshalNestedObject	2307
6.502.2.34	unmarshal	2308
6.502.3	Field Documentation	2308
6.502.3.1	DEFAULT_VERSION	2308
6.502.3.2	NULL_TYPE	2308
6.503	activemq::wireformat::openwire::OpenWireFormatFactory Class Reference	2308
6.503.1	Constructor & Destructor Documentation	2309
6.503.1.1	OpenWireFormatFactory	2309
6.503.1.2	~OpenWireFormatFactory	2309
6.503.2	Member Function Documentation	2309
6.503.2.1	createWireFormat	2309
6.504	activemq::wireformat::openwire::OpenWireFormatNegotiator Class Reference	2310
6.504.1	Constructor & Destructor Documentation	2310

6.504.1.1	OpenWireFormatNegotiator	2310
6.504.1.2	~OpenWireFormatNegotiator	2311
6.504.2	Member Function Documentation	2311
6.504.2.1	close	2311
6.504.2.2	onCommand	2311
6.504.2.3	oneway	2311
6.504.2.4	onTransportException	2312
6.504.2.5	request	2312
6.504.2.6	request	2312
6.504.2.7	start	2313
6.505	activemq::wireformat::openwire::OpenWireResponseBuilder Class Reference	2313
6.505.1	Constructor & Destructor Documentation	2314
6.505.1.1	OpenWireResponseBuilder	2314
6.505.1.2	~OpenWireResponseBuilder	2314
6.505.2	Member Function Documentation	2314
6.505.2.1	buildIncomingCommands	2314
6.505.2.2	buildResponse	2314
6.506	activemq::wireformat::openwire::utils::OpenwireStringSupport Class Reference . . .	2315
6.506.1	Constructor & Destructor Documentation	2315
6.506.1.1	OpenwireStringSupport	2315
6.506.1.2	~OpenwireStringSupport	2315
6.506.2	Member Function Documentation	2315
6.506.2.1	readString	2315
6.506.2.2	writeString	2316
6.507	decaf::io::OutputStream Class Reference	2316
6.507.1	Detailed Description	2317
6.507.2	Constructor & Destructor Documentation	2317
6.507.2.1	~OutputStream	2317
6.507.3	Member Function Documentation	2317
6.507.3.1	flush	2317
6.507.3.2	write	2317
6.507.3.3	write	2317
6.507.3.4	write	2318
6.508	activemq::commands::PartialCommand Class Reference	2318
6.508.1	Constructor & Destructor Documentation	2320
6.508.1.1	PartialCommand	2320

6.508.1.2	PartialCommand	2320
6.508.1.3	~PartialCommand	2320
6.508.2	Member Function Documentation	2320
6.508.2.1	cloneDataStructure	2320
6.508.2.2	copyDataStructure	2320
6.508.2.3	equals	2320
6.508.2.4	getCommandId	2321
6.508.2.5	getData	2321
6.508.2.6	getData	2321
6.508.2.7	getDataStructureType	2321
6.508.2.8	operator=	2321
6.508.2.9	setCommandId	2321
6.508.2.10	setData	2321
6.508.2.11	toString	2321
6.508.3	Field Documentation	2322
6.508.3.1	commandId	2322
6.508.3.2	data	2322
6.508.3.3	ID_PARTIALCOMMAND	2322
6.509	activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller Class	
	Reference	2322
6.509.1	Detailed Description	2323
6.509.2	Constructor & Destructor Documentation	2323
6.509.2.1	PartialCommandMarshaller	2323
6.509.2.2	~PartialCommandMarshaller	2323
6.509.3	Member Function Documentation	2323
6.509.3.1	createObject	2323
6.509.3.2	getDataStructureType	2323
6.509.3.3	looseMarshal	2324
6.509.3.4	looseUnmarshal	2324
6.509.3.5	tightMarshal1	2325
6.509.3.6	tightMarshal2	2325
6.509.3.7	tightUnmarshal	2326
6.510	activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller Class	
	Reference	2326
6.510.1	Detailed Description	2327
6.510.2	Constructor & Destructor Documentation	2327
6.510.2.1	PartialCommandMarshaller	2327

6.510.2.2 ~PartialCommandMarshaller	2327
6.510.3 Member Function Documentation	2327
6.510.3.1 createObject	2327
6.510.3.2 getDataStructureType	2328
6.510.3.3 looseMarshal	2328
6.510.3.4 looseUnmarshal	2328
6.510.3.5 tightMarshal1	2329
6.510.3.6 tightMarshal2	2329
6.510.3.7 tightUnmarshal	2330
6.511activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller Class	
Reference	2330
6.511.1 Detailed Description	2331
6.511.2 Constructor & Destructor Documentation	2332
6.511.2.1 PartialCommandMarshaller	2332
6.511.2.2 ~PartialCommandMarshaller	2332
6.511.3 Member Function Documentation	2332
6.511.3.1 createObject	2332
6.511.3.2 getDataStructureType	2332
6.511.3.3 looseMarshal	2332
6.511.3.4 looseUnmarshal	2333
6.511.3.5 tightMarshal1	2333
6.511.3.6 tightMarshal2	2334
6.511.3.7 tightUnmarshal	2334
6.512activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller Class	
Reference	2335
6.512.1 Detailed Description	2336
6.512.2 Constructor & Destructor Documentation	2336
6.512.2.1 PartialCommandMarshaller	2336
6.512.2.2 ~PartialCommandMarshaller	2336
6.512.3 Member Function Documentation	2336
6.512.3.1 createObject	2336
6.512.3.2 getDataStructureType	2336
6.512.3.3 looseMarshal	2336
6.512.3.4 looseUnmarshal	2337
6.512.3.5 tightMarshal1	2337
6.512.3.6 tightMarshal2	2338
6.512.3.7 tightUnmarshal	2338

6.513	activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller	Class	
	Reference		2339
6.513.1	Detailed Description		2340
6.513.2	Constructor & Destructor Documentation		2340
	6.513.2.1 PartialCommandMarshaller		2340
	6.513.2.2 ~PartialCommandMarshaller		2340
6.513.3	Member Function Documentation		2340
	6.513.3.1 createObject		2340
	6.513.3.2 getDataStructureType		2340
	6.513.3.3 looseMarshal		2341
	6.513.3.4 looseUnmarshal		2341
	6.513.3.5 tightMarshal1		2341
	6.513.3.6 tightMarshal2		2342
	6.513.3.7 tightUnmarshal		2342
6.514	decaf::lang::Pointer< T, REFCOUNTER > Class Template Reference		2343
	6.514.1 Detailed Description		2345
	6.514.2 Member Typedef Documentation		2345
	6.514.2.1 CounterType		2345
	6.514.2.2 PointerType		2345
	6.514.2.3 ReferenceType		2345
	6.514.3 Constructor & Destructor Documentation		2345
	6.514.3.1 Pointer		2345
	6.514.3.2 Pointer		2345
	6.514.3.3 Pointer		2346
	6.514.3.4 Pointer		2346
	6.514.3.5 Pointer		2346
	6.514.3.6 Pointer		2346
	6.514.3.7 ~Pointer		2347
	6.514.4 Member Function Documentation		2347
	6.514.4.1 dynamicCast		2347
	6.514.4.2 get		2347
	6.514.4.3 operator!		2347
	6.514.4.4 operator!=		2347
	6.514.4.5 operator*		2347
	6.514.4.6 operator*		2347
	6.514.4.7 operator->		2348
	6.514.4.8 operator->		2348

6.514.4.9 operator=	2348
6.514.4.10 operator=	2349
6.514.4.11 operator==	2349
6.514.4.12 release	2349
6.514.4.13 reset	2349
6.514.4.14 staticCast	2349
6.514.4.15 swap	2349
6.514.5 Friends And Related Function Documentation	2350
6.514.5.1 operator!=	2350
6.514.5.2 operator!=	2350
6.514.5.3 operator==	2350
6.514.5.4 operator==	2350
6.515 decaf::lang::PointerComparator< T, R > Class Template Reference	2350
6.515.1 Detailed Description	2351
6.515.2 Member Function Documentation	2351
6.515.2.1 compare	2351
6.515.2.2 operator()	2351
6.516 activemq::cmsutil::PooledSession Class Reference	2351
6.516.1 Detailed Description	2354
6.516.2 Constructor & Destructor Documentation	2354
6.516.2.1 PooledSession	2354
6.516.2.2 ~PooledSession	2354
6.516.3 Member Function Documentation	2354
6.516.3.1 close	2354
6.516.3.2 commit	2354
6.516.3.3 createBrowser	2354
6.516.3.4 createBrowser	2355
6.516.3.5 createBytesMessage	2355
6.516.3.6 createBytesMessage	2355
6.516.3.7 createCachedConsumer	2356
6.516.3.8 createCachedProducer	2356
6.516.3.9 createConsumer	2357
6.516.3.10 createConsumer	2357
6.516.3.11 createConsumer	2358
6.516.3.12 createDurableConsumer	2358
6.516.3.13 createMapMessage	2359

6.516.3.14	createMessage	2359
6.516.3.15	createProducer	2359
6.516.3.16	createQueue	2359
6.516.3.17	createStreamMessage	2360
6.516.3.18	createTemporaryQueue	2360
6.516.3.19	createTemporaryTopic	2360
6.516.3.20	createTextMessage	2361
6.516.3.21	createTextMessage	2361
6.516.3.22	createTopic	2361
6.516.3.23	getAcknowledgeMode	2362
6.516.3.24	getSession	2362
6.516.3.25	getSession	2362
6.516.3.26	isTransacted	2362
6.516.3.27	recover	2363
6.516.3.28	rollback	2363
6.516.3.29	unsubscribe	2363
6.517	decaf::util::concurrent::PooledThread Class Reference	2364
6.517.1	Constructor & Destructor Documentation	2364
6.517.1.1	PooledThread	2364
6.517.1.2	~PooledThread	2365
6.517.2	Member Function Documentation	2365
6.517.2.1	getPooledThreadListener	2365
6.517.2.2	isBusy	2365
6.517.2.3	run	2365
6.517.2.4	setPooledThreadListener	2365
6.517.2.5	stop	2365
6.518	decaf::util::concurrent::PooledThreadListener Class Reference	2366
6.518.1	Detailed Description	2366
6.518.2	Constructor & Destructor Documentation	2367
6.518.2.1	~PooledThreadListener	2367
6.518.3	Member Function Documentation	2367
6.518.3.1	onTaskCompleted	2367
6.518.3.2	onTaskException	2367
6.518.3.3	onTaskStarted	2367
6.519	decaf::net::PortUnreachableException Class Reference	2368
6.519.1	Constructor & Destructor Documentation	2368

6.519.1.1 PortUnreachableException	2368
6.519.1.2 PortUnreachableException	2368
6.519.1.3 PortUnreachableException	2369
6.519.1.4 PortUnreachableException	2369
6.519.1.5 PortUnreachableException	2369
6.519.1.6 PortUnreachableException	2369
6.519.1.7 ~PortUnreachableException	2370
6.519.2 Member Function Documentation	2370
6.519.2.1 clone	2370
6.520activemq::util::PrimitiveList Class Reference	2370
6.520.1 Detailed Description	2373
6.520.2 Constructor & Destructor Documentation	2373
6.520.2.1 PrimitiveList	2373
6.520.2.2 ~PrimitiveList	2373
6.520.2.3 PrimitiveList	2373
6.520.2.4 PrimitiveList	2373
6.520.3 Member Function Documentation	2373
6.520.3.1 getBool	2373
6.520.3.2 getByte	2374
6.520.3.3 getByteArray	2374
6.520.3.4 getChar	2375
6.520.3.5 getDouble	2375
6.520.3.6 getFloat	2375
6.520.3.7 getInt	2376
6.520.3.8 getLong	2376
6.520.3.9 getShort	2376
6.520.3.10getString	2377
6.520.3.11setBool	2377
6.520.3.12setByte	2378
6.520.3.13setByteArray	2378
6.520.3.14setChar	2378
6.520.3.15setDouble	2379
6.520.3.16setFloat	2379
6.520.3.17setInt	2379
6.520.3.18setLong	2380
6.520.3.19setShort	2380

6.520.3.20	getString	2380
6.520.3.21	toString	2381
6.521	activemq::util::PrimitiveMap Class Reference	2381
6.521.1	Detailed Description	2383
6.521.2	Constructor & Destructor Documentation	2383
6.521.2.1	PrimitiveMap	2383
6.521.2.2	~PrimitiveMap	2383
6.521.2.3	PrimitiveMap	2383
6.521.2.4	PrimitiveMap	2384
6.521.3	Member Function Documentation	2384
6.521.3.1	getBool	2384
6.521.3.2	getByte	2384
6.521.3.3	getByteArray	2385
6.521.3.4	getChar	2385
6.521.3.5	getDouble	2385
6.521.3.6	getFloat	2386
6.521.3.7	getInt	2386
6.521.3.8	getLong	2387
6.521.3.9	getShort	2387
6.521.3.10	getString	2387
6.521.3.11	setBool	2388
6.521.3.12	setByte	2388
6.521.3.13	setByteArray	2388
6.521.3.14	setChar	2388
6.521.3.15	setDouble	2389
6.521.3.16	setFloat	2389
6.521.3.17	setInt	2389
6.521.3.18	setLong	2389
6.521.3.19	setShort	2390
6.521.3.20	getString	2390
6.521.3.21	toString	2390
6.522	activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller Class Reference	2390
6.522.1	Detailed Description	2392
6.522.2	Constructor & Destructor Documentation	2392
6.522.2.1	PrimitiveTypesMarshaller	2392
6.522.2.2	~PrimitiveTypesMarshaller	2392

6.522.3 Member Function Documentation	2392
6.522.3.1 marshal	2392
6.522.3.2 marshal	2392
6.522.3.3 marshalPrimitive	2393
6.522.3.4 marshalPrimitiveList	2393
6.522.3.5 marshalPrimitiveMap	2393
6.522.3.6 unmarshal	2394
6.522.3.7 unmarshal	2394
6.522.3.8 unmarshalPrimitive	2394
6.522.3.9 unmarshalPrimitiveList	2395
6.522.3.10unmarshalPrimitiveMap	2395
6.523activemq::util::PrimitiveValueNode::PrimitiveValue Union Reference	2395
6.523.1 Detailed Description	2396
6.523.2 Field Documentation	2396
6.523.2.1 boolValue	2396
6.523.2.2 byteArrayValue	2396
6.523.2.3 byteValue	2396
6.523.2.4 charValue	2396
6.523.2.5 doubleValue	2396
6.523.2.6 float Value	2396
6.523.2.7 int Value	2396
6.523.2.8 list Value	2396
6.523.2.9 longValue	2396
6.523.2.10mapValue	2396
6.523.2.11shortValue	2396
6.523.2.12stringValue	2396
6.524activemq::util::PrimitiveValueConverter Class Reference	2397
6.524.1 Detailed Description	2397
6.524.2 Constructor & Destructor Documentation	2397
6.524.2.1 PrimitiveValueConverter	2397
6.524.2.2 ~PrimitiveValueConverter	2397
6.524.3 Member Function Documentation	2397
6.524.3.1 convert	2397
6.525activemq::util::PrimitiveValueNode Class Reference	2398
6.525.1 Detailed Description	2401
6.525.2 Member Enumeration Documentation	2401

6.525.2.1 PrimitiveType	2401
6.525.3 Constructor & Destructor Documentation	2402
6.525.3.1 PrimitiveValueNode	2402
6.525.3.2 PrimitiveValueNode	2402
6.525.3.3 PrimitiveValueNode	2402
6.525.3.4 PrimitiveValueNode	2402
6.525.3.5 PrimitiveValueNode	2403
6.525.3.6 PrimitiveValueNode	2403
6.525.3.7 PrimitiveValueNode	2403
6.525.3.8 PrimitiveValueNode	2403
6.525.3.9 PrimitiveValueNode	2403
6.525.3.10 PrimitiveValueNode	2403
6.525.3.11 PrimitiveValueNode	2404
6.525.3.12 PrimitiveValueNode	2404
6.525.3.13 PrimitiveValueNode	2404
6.525.3.14 PrimitiveValueNode	2404
6.525.3.15 PrimitiveValueNode	2404
6.525.3.16 ~PrimitiveValueNode	2405
6.525.4 Member Function Documentation	2405
6.525.4.1 clear	2405
6.525.4.2 getBool	2405
6.525.4.3 getByte	2405
6.525.4.4 getByteArray	2405
6.525.4.5 getChar	2406
6.525.4.6 getDouble	2406
6.525.4.7 getFloat	2406
6.525.4.8 getInt	2406
6.525.4.9 getList	2407
6.525.4.10 getLong	2407
6.525.4.11 getMap	2407
6.525.4.12 getShort	2407
6.525.4.13 getString	2408
6.525.4.14 getType	2408
6.525.4.15 getValue	2408
6.525.4.16 operator=	2408
6.525.4.17 operator==	2408

6.525.4.18	setBool	2409
6.525.4.19	setByte	2409
6.525.4.20	setByteArray	2409
6.525.4.21	setChar	2409
6.525.4.22	setDouble	2409
6.525.4.23	setFloat	2410
6.525.4.24	setInt	2410
6.525.4.25	setList	2410
6.525.4.26	setLong	2410
6.525.4.27	setMap	2410
6.525.4.28	setShort	2411
6.525.4.29	setString	2411
6.525.4.30	setValue	2411
6.525.4.31	toString	2411
6.526	decaf::security::Principal Class Reference	2411
6.526.1	Detailed Description	2412
6.526.2	Constructor & Destructor Documentation	2412
6.526.2.1	~Principal	2412
6.526.3	Member Function Documentation	2412
6.526.3.1	equals	2412
6.526.3.2	getName	2412
6.527	decaf::util::PriorityQueue< E > Class Template Reference	2413
6.527.1	Detailed Description	2414
6.527.2	Constructor & Destructor Documentation	2415
6.527.2.1	PriorityQueue	2415
6.527.2.2	PriorityQueue	2415
6.527.2.3	PriorityQueue	2415
6.527.2.4	PriorityQueue	2416
6.527.2.5	PriorityQueue	2416
6.527.2.6	~PriorityQueue	2416
6.527.3	Member Function Documentation	2416
6.527.3.1	add	2416
6.527.3.2	clear	2417
6.527.3.3	comparator	2417
6.527.3.4	iterator	2417
6.527.3.5	iterator	2417

6.527.3.6 offer	2417
6.527.3.7 operator=	2418
6.527.3.8 operator=	2418
6.527.3.9 peek	2418
6.527.3.10 poll	2418
6.527.3.11 remove	2419
6.527.3.12 remove	2419
6.527.3.13 size	2420
6.527.4 Friends And Related Function Documentation	2420
6.527.4.1 PriorityQueueIterator	2420
6.528 activemq::commands::ProducerAck Class Reference	2420
6.528.1 Constructor & Destructor Documentation	2421
6.528.1.1 ProducerAck	2421
6.528.1.2 ProducerAck	2421
6.528.1.3 ~ProducerAck	2421
6.528.2 Member Function Documentation	2421
6.528.2.1 cloneDataStructure	2421
6.528.2.2 copyDataStructure	2422
6.528.2.3 equals	2422
6.528.2.4 getDataStructureType	2422
6.528.2.5 getProducerId	2422
6.528.2.6 getProducerId	2422
6.528.2.7 getSize	2422
6.528.2.8 isProducerAck	2422
6.528.2.9 operator=	2423
6.528.2.10 setProducerId	2423
6.528.2.11 setSize	2423
6.528.2.12 toString	2423
6.528.2.13 visit	2423
6.528.3 Field Documentation	2423
6.528.3.1 ID_PRODUCERACK	2423
6.528.3.2 producerId	2423
6.528.3.3 size	2423
6.529 activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller Class Reference	2424
6.529.1 Detailed Description	2424
6.529.2 Constructor & Destructor Documentation	2425

6.529.2.1	ProducerAckMarshaller	2425
6.529.2.2	~ProducerAckMarshaller	2425
6.529.3	Member Function Documentation	2425
6.529.3.1	createObject	2425
6.529.3.2	getDataStructureType	2425
6.529.3.3	looseMarshal	2425
6.529.3.4	looseUnmarshal	2426
6.529.3.5	tightMarshal1	2426
6.529.3.6	tightMarshal2	2426
6.529.3.7	tightUnmarshal	2427
6.530	activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller Class Reference	2427
6.530.1	Detailed Description	2428
6.530.2	Constructor & Destructor Documentation	2429
6.530.2.1	ProducerAckMarshaller	2429
6.530.2.2	~ProducerAckMarshaller	2429
6.530.3	Member Function Documentation	2429
6.530.3.1	createObject	2429
6.530.3.2	getDataStructureType	2429
6.530.3.3	looseMarshal	2429
6.530.3.4	looseUnmarshal	2430
6.530.3.5	tightMarshal1	2430
6.530.3.6	tightMarshal2	2430
6.530.3.7	tightUnmarshal	2431
6.531	activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller Class Reference	2431
6.531.1	Detailed Description	2432
6.531.2	Constructor & Destructor Documentation	2433
6.531.2.1	ProducerAckMarshaller	2433
6.531.2.2	~ProducerAckMarshaller	2433
6.531.3	Member Function Documentation	2433
6.531.3.1	createObject	2433
6.531.3.2	getDataStructureType	2433
6.531.3.3	looseMarshal	2433
6.531.3.4	looseUnmarshal	2434
6.531.3.5	tightMarshal1	2434
6.531.3.6	tightMarshal2	2434
6.531.3.7	tightUnmarshal	2435

6.532	activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller Class Reference	2435
6.532.1	Detailed Description	2436
6.532.2	Constructor & Destructor Documentation	2437
6.532.2.1	ProducerAckMarshaller	2437
6.532.2.2	~ProducerAckMarshaller	2437
6.532.3	Member Function Documentation	2437
6.532.3.1	createObject	2437
6.532.3.2	getDataStructureType	2437
6.532.3.3	looseMarshal	2437
6.532.3.4	looseUnmarshal	2438
6.532.3.5	tightMarshal1	2438
6.532.3.6	tightMarshal2	2438
6.532.3.7	tightUnmarshal	2439
6.533	activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller Class Reference	2439
6.533.1	Detailed Description	2440
6.533.2	Constructor & Destructor Documentation	2441
6.533.2.1	ProducerAckMarshaller	2441
6.533.2.2	~ProducerAckMarshaller	2441
6.533.3	Member Function Documentation	2441
6.533.3.1	createObject	2441
6.533.3.2	getDataStructureType	2441
6.533.3.3	looseMarshal	2441
6.533.3.4	looseUnmarshal	2442
6.533.3.5	tightMarshal1	2442
6.533.3.6	tightMarshal2	2442
6.533.3.7	tightUnmarshal	2443
6.534	activemq::cmsutil::ProducerCallback Class Reference	2443
6.534.1	Detailed Description	2444
6.534.2	Constructor & Destructor Documentation	2444
6.534.2.1	~ProducerCallback	2444
6.534.3	Member Function Documentation	2444
6.534.3.1	doInCms	2444
6.535	activemq::cmsutil::CmsTemplate::ProducerExecutor Class Reference	2444
6.535.1	Constructor & Destructor Documentation	2445
6.535.1.1	ProducerExecutor	2445
6.535.1.2	~ProducerExecutor	2445

6.535.2 Member Function Documentation	2445
6.535.2.1 doInCms	2445
6.535.2.2 getDestination	2446
6.535.3 Field Documentation	2446
6.535.3.1 action	2446
6.535.3.2 destination	2446
6.535.3.3 parent	2446
6.536activemq::commands::ProducerId Class Reference	2446
6.536.1 Member Typedef Documentation	2447
6.536.1.1 COMPARATOR	2447
6.536.2 Constructor & Destructor Documentation	2447
6.536.2.1 ProducerId	2447
6.536.2.2 ProducerId	2447
6.536.2.3 ProducerId	2447
6.536.2.4 ~ProducerId	2448
6.536.3 Member Function Documentation	2448
6.536.3.1 cloneDataStructure	2448
6.536.3.2 compareTo	2448
6.536.3.3 copyDataStructure	2448
6.536.3.4 equals	2448
6.536.3.5 equals	2448
6.536.3.6 getConnectionId	2448
6.536.3.7 getConnectionId	2448
6.536.3.8 getDataStructureType	2448
6.536.3.9 getParentId	2449
6.536.3.10getSessionId	2449
6.536.3.11getValue	2449
6.536.3.12operator<	2449
6.536.3.13operator=	2449
6.536.3.14operator==	2449
6.536.3.15setConnectionId	2449
6.536.3.16setSessionId	2449
6.536.3.17setValue	2449
6.536.3.18oString	2449
6.536.4 Field Documentation	2449
6.536.4.1 connectionId	2449

6.536.4.2 ID_PRODUCERID	2449
6.536.4.3 sessionId	2450
6.536.4.4 value	2450
6.537activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller Class Reference	2450
6.537.1 Detailed Description	2451
6.537.2 Constructor & Destructor Documentation	2451
6.537.2.1 ProducerIdMarshaller	2451
6.537.2.2 ~ProducerIdMarshaller	2451
6.537.3 Member Function Documentation	2451
6.537.3.1 createObject	2451
6.537.3.2 getDataStructureType	2451
6.537.3.3 looseMarshal	2452
6.537.3.4 looseUnmarshal	2452
6.537.3.5 tightMarshal1	2452
6.537.3.6 tightMarshal2	2453
6.537.3.7 tightUnmarshal	2453
6.538activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller Class Reference	2454
6.538.1 Detailed Description	2455
6.538.2 Constructor & Destructor Documentation	2455
6.538.2.1 ProducerIdMarshaller	2455
6.538.2.2 ~ProducerIdMarshaller	2455
6.538.3 Member Function Documentation	2455
6.538.3.1 createObject	2455
6.538.3.2 getDataStructureType	2455
6.538.3.3 looseMarshal	2455
6.538.3.4 looseUnmarshal	2456
6.538.3.5 tightMarshal1	2456
6.538.3.6 tightMarshal2	2457
6.538.3.7 tightUnmarshal	2457
6.539activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller Class Reference	2457
6.539.1 Detailed Description	2458
6.539.2 Constructor & Destructor Documentation	2459
6.539.2.1 ProducerIdMarshaller	2459
6.539.2.2 ~ProducerIdMarshaller	2459
6.539.3 Member Function Documentation	2459
6.539.3.1 createObject	2459

6.539.3.2	getDataStructureType	2459
6.539.3.3	looseMarshal	2459
6.539.3.4	looseUnmarshal	2460
6.539.3.5	tightMarshal1	2460
6.539.3.6	tightMarshal2	2460
6.539.3.7	tightUnmarshal	2461
6.540	activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller Class Reference	2461
6.540.1	Detailed Description	2462
6.540.2	Constructor & Destructor Documentation	2463
6.540.2.1	ProducerIdMarshaller	2463
6.540.2.2	~ProducerIdMarshaller	2463
6.540.3	Member Function Documentation	2463
6.540.3.1	createObject	2463
6.540.3.2	getDataStructureType	2463
6.540.3.3	looseMarshal	2463
6.540.3.4	looseUnmarshal	2464
6.540.3.5	tightMarshal1	2464
6.540.3.6	tightMarshal2	2464
6.540.3.7	tightUnmarshal	2465
6.541	activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller Class Reference	2465
6.541.1	Detailed Description	2466
6.541.2	Constructor & Destructor Documentation	2467
6.541.2.1	ProducerIdMarshaller	2467
6.541.2.2	~ProducerIdMarshaller	2467
6.541.3	Member Function Documentation	2467
6.541.3.1	createObject	2467
6.541.3.2	getDataStructureType	2467
6.541.3.3	looseMarshal	2467
6.541.3.4	looseUnmarshal	2468
6.541.3.5	tightMarshal1	2468
6.541.3.6	tightMarshal2	2468
6.541.3.7	tightUnmarshal	2469
6.542	activemq::commands::ProducerInfo Class Reference	2469
6.542.1	Constructor & Destructor Documentation	2471
6.542.1.1	ProducerInfo	2471
6.542.1.2	ProducerInfo	2471

6.542.1.3 ~ProducerInfo	2471
6.542.2 Member Function Documentation	2471
6.542.2.1 cloneDataStructure	2471
6.542.2.2 copyDataStructure	2471
6.542.2.3 equals	2471
6.542.2.4 getBrokerPath	2472
6.542.2.5 getBrokerPath	2472
6.542.2.6 getDataStructureType	2472
6.542.2.7 getDestination	2472
6.542.2.8 getDestination	2472
6.542.2.9 getProducerId	2472
6.542.2.10 getProducerId	2472
6.542.2.11 getWindowSize	2472
6.542.2.12 sDispatchAsync	2472
6.542.2.13 sProducerInfo	2472
6.542.2.14 operator=	2473
6.542.2.15 setBrokerPath	2473
6.542.2.16 setDestination	2473
6.542.2.17 setDispatchAsync	2473
6.542.2.18 setProducerId	2473
6.542.2.19 setWindowSize	2473
6.542.2.20 toString	2473
6.542.2.21 visit	2473
6.542.3 Field Documentation	2474
6.542.3.1 brokerPath	2474
6.542.3.2 destination	2474
6.542.3.3 dispatchAsync	2474
6.542.3.4 ID_PRODUCERINFO	2474
6.542.3.5 producerId	2474
6.542.3.6 windowSize	2474
6.543 activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller Class Reference	2474
6.543.1 Detailed Description	2475
6.543.2 Constructor & Destructor Documentation	2475
6.543.2.1 ProducerInfoMarshaller	2475
6.543.2.2 ~ProducerInfoMarshaller	2475
6.543.3 Member Function Documentation	2475

6.543.3.1 createObject	2475
6.543.3.2 getDataStructureType	2475
6.543.3.3 looseMarshal	2476
6.543.3.4 looseUnmarshal	2476
6.543.3.5 tightMarshal1	2476
6.543.3.6 tightMarshal2	2477
6.543.3.7 tightUnmarshal	2477
6.544activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller Class Reference	2478
6.544.1 Detailed Description	2479
6.544.2 Constructor & Destructor Documentation	2479
6.544.2.1 ProducerInfoMarshaller	2479
6.544.2.2 ~ProducerInfoMarshaller	2479
6.544.3 Member Function Documentation	2479
6.544.3.1 createObject	2479
6.544.3.2 getDataStructureType	2479
6.544.3.3 looseMarshal	2480
6.544.3.4 looseUnmarshal	2480
6.544.3.5 tightMarshal1	2480
6.544.3.6 tightMarshal2	2481
6.544.3.7 tightUnmarshal	2481
6.545activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller Class Reference	2482
6.545.1 Detailed Description	2483
6.545.2 Constructor & Destructor Documentation	2483
6.545.2.1 ProducerInfoMarshaller	2483
6.545.2.2 ~ProducerInfoMarshaller	2483
6.545.3 Member Function Documentation	2483
6.545.3.1 createObject	2483
6.545.3.2 getDataStructureType	2483
6.545.3.3 looseMarshal	2484
6.545.3.4 looseUnmarshal	2484
6.545.3.5 tightMarshal1	2484
6.545.3.6 tightMarshal2	2485
6.545.3.7 tightUnmarshal	2485
6.546activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller Class Reference	2486

6.546.1 Detailed Description	2487
6.546.2 Constructor & Destructor Documentation	2487
6.546.2.1 ProducerInfoMarshaller	2487
6.546.2.2 ~ProducerInfoMarshaller	2487
6.546.3 Member Function Documentation	2487
6.546.3.1 createObject	2487
6.546.3.2 getDataStructureType	2487
6.546.3.3 looseMarshal	2488
6.546.3.4 looseUnmarshal	2488
6.546.3.5 tightMarshal1	2488
6.546.3.6 tightMarshal2	2489
6.546.3.7 tightUnmarshal	2489
6.547activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller Class Reference	2490
6.547.1 Detailed Description	2491
6.547.2 Constructor & Destructor Documentation	2491
6.547.2.1 ProducerInfoMarshaller	2491
6.547.2.2 ~ProducerInfoMarshaller	2491
6.547.3 Member Function Documentation	2491
6.547.3.1 createObject	2491
6.547.3.2 getDataStructureType	2491
6.547.3.3 looseMarshal	2492
6.547.3.4 looseUnmarshal	2492
6.547.3.5 tightMarshal1	2492
6.547.3.6 tightMarshal2	2493
6.547.3.7 tightUnmarshal	2493
6.548activemq::state::ProducerState Class Reference	2494
6.548.1 Constructor & Destructor Documentation	2494
6.548.1.1 ProducerState	2494
6.548.1.2 ~ProducerState	2494
6.548.2 Member Function Documentation	2494
6.548.2.1 getInfo	2494
6.548.2.2 toString	2494
6.549decaf::util::Properties Class Reference	2494
6.549.1 Detailed Description	2496
6.549.2 Constructor & Destructor Documentation	2496
6.549.2.1 Properties	2496

6.549.2.2 Properties	2496
6.549.2.3 ~Properties	2496
6.549.3 Member Function Documentation	2496
6.549.3.1 clear	2496
6.549.3.2 clone	2497
6.549.3.3 copy	2497
6.549.3.4 equals	2497
6.549.3.5 getProperty	2497
6.549.3.6 getProperty	2498
6.549.3.7 hasProperty	2498
6.549.3.8 isEmpty	2498
6.549.3.9 load	2498
6.549.3.10load	2500
6.549.3.11operator=	2500
6.549.3.12remove	2501
6.549.3.13setProperty	2501
6.549.3.14size	2501
6.549.3.15store	2501
6.549.3.16store	2502
6.549.3.17toArray	2502
6.549.3.18toString	2503
6.549.4 Field Documentation	2503
6.549.4.1 defaults	2503
6.550decaf::util::logging::PropertiesChangeListener Class Reference	2503
6.550.1 Detailed Description	2503
6.550.2 Constructor & Destructor Documentation	2504
6.550.2.1 ~PropertiesChangeListener	2504
6.550.3 Member Function Documentation	2504
6.550.3.1 onPropertyChanged	2504
6.551decaf::net::ProtocolException Class Reference	2504
6.551.1 Constructor & Destructor Documentation	2505
6.551.1.1 ProtocolException	2505
6.551.1.2 ProtocolException	2505
6.551.1.3 ProtocolException	2505
6.551.1.4 ProtocolException	2505
6.551.1.5 ProtocolException	2506

6.551.1.6 ProtocolException	2506
6.551.1.7 ~ProtocolException	2506
6.551.2 Member Function Documentation	2506
6.551.2.1 clone	2506
6.552decaf::security::PublicKey Class Reference	2506
6.552.1 Detailed Description	2507
6.552.2 Constructor & Destructor Documentation	2507
6.552.2.1 ~PublicKey	2507
6.553decaf::util::Queue< E > Class Template Reference	2507
6.553.1 Detailed Description	2508
6.553.2 Constructor & Destructor Documentation	2508
6.553.2.1 ~Queue	2508
6.553.3 Member Function Documentation	2508
6.553.3.1 element	2508
6.553.3.2 offer	2509
6.553.3.3 peek	2509
6.553.3.4 poll	2509
6.553.3.5 remove	2510
6.554cms::Queue Class Reference	2510
6.554.1 Detailed Description	2511
6.554.2 Constructor & Destructor Documentation	2511
6.554.2.1 ~Queue	2511
6.554.3 Member Function Documentation	2511
6.554.3.1 getQueueName	2511
6.555cms::QueueBrowser Class Reference	2511
6.555.1 Detailed Description	2512
6.555.2 Constructor & Destructor Documentation	2512
6.555.2.1 ~QueueBrowser	2512
6.555.3 Member Function Documentation	2512
6.555.3.1 getEnumeration	2512
6.555.3.2 getMessageSelector	2512
6.555.3.3 getQueue	2513
6.556decaf::util::Random Class Reference	2513
6.556.1 Detailed Description	2514
6.556.2 Constructor & Destructor Documentation	2514
6.556.2.1 Random	2514

6.556.2.2 Random	2514
6.556.3 Member Function Documentation	2515
6.556.3.1 next	2515
6.556.3.2 nextBoolean	2515
6.556.3.3 nextBytes	2515
6.556.3.4 nextDouble	2515
6.556.3.5 nextFloat	2516
6.556.3.6 nextGaussian	2516
6.556.3.7 nextInt	2516
6.556.3.8 nextInt	2517
6.556.3.9 nextLong	2517
6.556.3.10 setSeed	2517
6.557 activemq::transport::inactivity::ReadChecker Class Reference	2517
6.557.1 Detailed Description	2518
6.557.2 Constructor & Destructor Documentation	2518
6.557.2.1 ReadChecker	2518
6.557.2.2 ~ReadChecker	2518
6.557.3 Member Function Documentation	2518
6.557.3.1 run	2518
6.558 decaf::io::Reader Class Reference	2518
6.558.1 Constructor & Destructor Documentation	2519
6.558.1.1 ~Reader	2519
6.558.2 Member Function Documentation	2519
6.558.2.1 getInputStream	2519
6.558.2.2 read	2519
6.558.2.3 readByte	2519
6.558.2.4 setInputStream	2520
6.559 decaf::nio::ReadOnlyBufferException Class Reference	2520
6.559.1 Constructor & Destructor Documentation	2520
6.559.1.1 ReadOnlyBufferException	2520
6.559.1.2 ReadOnlyBufferException	2521
6.559.1.3 ReadOnlyBufferException	2521
6.559.1.4 ReadOnlyBufferException	2521
6.559.1.5 ReadOnlyBufferException	2521
6.559.1.6 ReadOnlyBufferException	2521
6.559.1.7 ~ReadOnlyBufferException	2522

6.559.2 Member Function Documentation	2522
6.559.2.1 clone	2522
6.560decaf::util::concurrent::locks::ReadWriteLock Class Reference	2522
6.560.1 Detailed Description	2522
6.560.2 Constructor & Destructor Documentation	2524
6.560.2.1 ~ReadWriteLock	2524
6.560.3 Member Function Documentation	2524
6.560.3.1 readLock	2524
6.560.3.2 writeLock	2524
6.561activemq::cmsutil::CmsTemplate::ReceiveExecutor Class Reference	2524
6.561.1 Constructor & Destructor Documentation	2525
6.561.1.1 ReceiveExecutor	2525
6.561.1.2 ~ReceiveExecutor	2525
6.561.2 Member Function Documentation	2525
6.561.2.1 doInCms	2525
6.561.2.2 getDestination	2526
6.561.2.3 getMessage	2526
6.561.3 Field Documentation	2526
6.561.3.1 destination	2526
6.561.3.2 message	2526
6.561.3.3 noLocal	2526
6.561.3.4 parent	2526
6.561.3.5 selector	2526
6.562decaf::util::concurrent::locks::ReentrantLock Class Reference	2526
6.562.1 Detailed Description	2527
6.562.2 Constructor & Destructor Documentation	2528
6.562.2.1 ReentrantLock	2528
6.562.2.2 ~ReentrantLock	2528
6.562.3 Member Function Documentation	2528
6.562.3.1 getHoldCount	2528
6.562.3.2 isFair	2529
6.562.3.3 isHeldByCurrentThread	2529
6.562.3.4 isLocked	2529
6.562.3.5 lock	2529
6.562.3.6 lockInterruptibly	2530
6.562.3.7 newCondition	2530

6.562.3.8 toString	2531
6.562.3.9 tryLock	2531
6.562.3.10 tryLock	2532
6.562.3.11 unlock	2533
6.563 decaf::util::concurrent::RejectedExecutionException Class Reference	2533
6.563.1 Constructor & Destructor Documentation	2534
6.563.1.1 RejectedExecutionException	2534
6.563.1.2 RejectedExecutionException	2534
6.563.1.3 RejectedExecutionException	2534
6.563.1.4 RejectedExecutionException	2534
6.563.1.5 RejectedExecutionException	2534
6.563.1.6 RejectedExecutionException	2535
6.563.1.7 ~RejectedExecutionException	2535
6.563.2 Member Function Documentation	2535
6.563.2.1 clone	2535
6.564 decaf::util::concurrent::RejectedExecutionHandler Class Reference	2536
6.564.1 Detailed Description	2536
6.564.2 Constructor & Destructor Documentation	2536
6.564.2.1 ~RejectedExecutionHandler	2536
6.565 activemq::commands::RemoveInfo Class Reference	2536
6.565.1 Constructor & Destructor Documentation	2538
6.565.1.1 RemoveInfo	2538
6.565.1.2 RemoveInfo	2538
6.565.1.3 ~RemoveInfo	2538
6.565.2 Member Function Documentation	2538
6.565.2.1 cloneDataStructure	2538
6.565.2.2 copyDataStructure	2538
6.565.2.3 equals	2538
6.565.2.4 getDataStructureType	2538
6.565.2.5 getLastDeliveredSequenceId	2539
6.565.2.6 getObjectId	2539
6.565.2.7 getObjectId	2539
6.565.2.8 isRemoveInfo	2539
6.565.2.9 operator=	2539
6.565.2.10 setLastDeliveredSequenceId	2539
6.565.2.11 setObjectId	2539

6.565.2.12	toString	2539
6.565.2.13	visit	2540
6.565.3	Field Documentation	2540
6.565.3.1	ID_REMOVEINFO	2540
6.565.3.2	lastDeliveredSequenceId	2540
6.565.3.3	objectId	2540
6.566	activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller Class Reference	2540
6.566.1	Detailed Description	2541
6.566.2	Constructor & Destructor Documentation	2541
6.566.2.1	RemoveInfoMarshaller	2541
6.566.2.2	~RemoveInfoMarshaller	2541
6.566.3	Member Function Documentation	2541
6.566.3.1	createObject	2541
6.566.3.2	getDataStructureType	2542
6.566.3.3	looseMarshal	2542
6.566.3.4	looseUnmarshal	2542
6.566.3.5	tightMarshal1	2543
6.566.3.6	tightMarshal2	2543
6.566.3.7	tightUnmarshal	2543
6.567	activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller Class Reference	2544
6.567.1	Detailed Description	2545
6.567.2	Constructor & Destructor Documentation	2545
6.567.2.1	RemoveInfoMarshaller	2545
6.567.2.2	~RemoveInfoMarshaller	2545
6.567.3	Member Function Documentation	2545
6.567.3.1	createObject	2545
6.567.3.2	getDataStructureType	2545
6.567.3.3	looseMarshal	2546
6.567.3.4	looseUnmarshal	2546
6.567.3.5	tightMarshal1	2546
6.567.3.6	tightMarshal2	2547
6.567.3.7	tightUnmarshal	2547
6.568	activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller Class Reference	2548
6.568.1	Detailed Description	2549
6.568.2	Constructor & Destructor Documentation	2549
6.568.2.1	RemoveInfoMarshaller	2549

6.568.2.2 ~RemoveInfoMarshaller	2549
6.568.3 Member Function Documentation	2549
6.568.3.1 createObject	2549
6.568.3.2 getDataStructureType	2549
6.568.3.3 looseMarshal	2550
6.568.3.4 looseUnmarshal	2550
6.568.3.5 tightMarshal1	2550
6.568.3.6 tightMarshal2	2551
6.568.3.7 tightUnmarshal	2551
6.569activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller Class Reference	2552
6.569.1 Detailed Description	2553
6.569.2 Constructor & Destructor Documentation	2553
6.569.2.1 RemoveInfoMarshaller	2553
6.569.2.2 ~RemoveInfoMarshaller	2553
6.569.3 Member Function Documentation	2553
6.569.3.1 createObject	2553
6.569.3.2 getDataStructureType	2553
6.569.3.3 looseMarshal	2554
6.569.3.4 looseUnmarshal	2554
6.569.3.5 tightMarshal1	2554
6.569.3.6 tightMarshal2	2555
6.569.3.7 tightUnmarshal	2555
6.570activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller Class Reference	2556
6.570.1 Detailed Description	2557
6.570.2 Constructor & Destructor Documentation	2557
6.570.2.1 RemoveInfoMarshaller	2557
6.570.2.2 ~RemoveInfoMarshaller	2557
6.570.3 Member Function Documentation	2557
6.570.3.1 createObject	2557
6.570.3.2 getDataStructureType	2557
6.570.3.3 looseMarshal	2558
6.570.3.4 looseUnmarshal	2558
6.570.3.5 tightMarshal1	2558
6.570.3.6 tightMarshal2	2559
6.570.3.7 tightUnmarshal	2559
6.571activemq::commands::RemoveSubscriptionInfo Class Reference	2560

6.571.1 Constructor & Destructor Documentation	2561
6.571.1.1 RemoveSubscriptionInfo	2561
6.571.1.2 RemoveSubscriptionInfo	2561
6.571.1.3 ~RemoveSubscriptionInfo	2561
6.571.2 Member Function Documentation	2561
6.571.2.1 cloneDataStructure	2561
6.571.2.2 copyDataStructure	2562
6.571.2.3 equals	2562
6.571.2.4 getClientId	2562
6.571.2.5 getClientId	2562
6.571.2.6 getConnectionId	2562
6.571.2.7 getConnectionId	2562
6.571.2.8 getDataStructureType	2562
6.571.2.9 getSubscriptionName	2563
6.571.2.10 getSubscriptionName	2563
6.571.2.11 isRemoveSubscriptionInfo	2563
6.571.2.12 operator=	2563
6.571.2.13 setClientId	2563
6.571.2.14 setConnectionId	2563
6.571.2.15 setSubscriptionName	2563
6.571.2.16 toString	2563
6.571.2.17 visit	2563
6.571.3 Field Documentation	2564
6.571.3.1 clientId	2564
6.571.3.2 connectionId	2564
6.571.3.3 ID_REMOVESUBSCRIPTIONINFO	2564
6.571.3.4 subscriptionName	2564
6.572 activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller	
Class Reference	2564
6.572.1 Detailed Description	2565
6.572.2 Constructor & Destructor Documentation	2565
6.572.2.1 RemoveSubscriptionInfoMarshaller	2565
6.572.2.2 ~RemoveSubscriptionInfoMarshaller	2565
6.572.3 Member Function Documentation	2565
6.572.3.1 createObject	2565
6.572.3.2 getDataStructureType	2566
6.572.3.3 looseMarshal	2566

6.572.3.4 looseUnmarshal	2566
6.572.3.5 tightMarshal1	2567
6.572.3.6 tightMarshal2	2567
6.572.3.7 tightUnmarshal	2567
6.573activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller	
Class Reference	2568
6.573.1 Detailed Description	2569
6.573.2 Constructor & Destructor Documentation	2569
6.573.2.1 RemoveSubscriptionInfoMarshaller	2569
6.573.2.2 ~RemoveSubscriptionInfoMarshaller	2569
6.573.3 Member Function Documentation	2569
6.573.3.1 createObject	2569
6.573.3.2 getDataStructureType	2569
6.573.3.3 looseMarshal	2570
6.573.3.4 looseUnmarshal	2570
6.573.3.5 tightMarshal1	2570
6.573.3.6 tightMarshal2	2571
6.573.3.7 tightUnmarshal	2571
6.574activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller	
Class Reference	2572
6.574.1 Detailed Description	2573
6.574.2 Constructor & Destructor Documentation	2573
6.574.2.1 RemoveSubscriptionInfoMarshaller	2573
6.574.2.2 ~RemoveSubscriptionInfoMarshaller	2573
6.574.3 Member Function Documentation	2573
6.574.3.1 createObject	2573
6.574.3.2 getDataStructureType	2573
6.574.3.3 looseMarshal	2574
6.574.3.4 looseUnmarshal	2574
6.574.3.5 tightMarshal1	2574
6.574.3.6 tightMarshal2	2575
6.574.3.7 tightUnmarshal	2575
6.575activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller	
Class Reference	2576
6.575.1 Detailed Description	2577
6.575.2 Constructor & Destructor Documentation	2577
6.575.2.1 RemoveSubscriptionInfoMarshaller	2577

6.575.2.2 ~RemoveSubscriptionInfoMarshaller	2577
6.575.3 Member Function Documentation	2577
6.575.3.1 createObject	2577
6.575.3.2 getDataStructureType	2577
6.575.3.3 looseMarshal	2578
6.575.3.4 looseUnmarshal	2578
6.575.3.5 tightMarshal1	2578
6.575.3.6 tightMarshal2	2579
6.575.3.7 tightUnmarshal	2579
6.576activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller	
Class Reference	2580
6.576.1 Detailed Description	2581
6.576.2 Constructor & Destructor Documentation	2581
6.576.2.1 RemoveSubscriptionInfoMarshaller	2581
6.576.2.2 ~RemoveSubscriptionInfoMarshaller	2581
6.576.3 Member Function Documentation	2581
6.576.3.1 createObject	2581
6.576.3.2 getDataStructureType	2581
6.576.3.3 looseMarshal	2582
6.576.3.4 looseUnmarshal	2582
6.576.3.5 tightMarshal1	2582
6.576.3.6 tightMarshal2	2583
6.576.3.7 tightUnmarshal	2583
6.577activemq::commands::ReplayCommand Class Reference	2584
6.577.1 Constructor & Destructor Documentation	2585
6.577.1.1 ReplayCommand	2585
6.577.1.2 ReplayCommand	2585
6.577.1.3 ~ReplayCommand	2585
6.577.2 Member Function Documentation	2585
6.577.2.1 cloneDataStructure	2585
6.577.2.2 copyDataStructure	2585
6.577.2.3 equals	2586
6.577.2.4 getDataStructureType	2586
6.577.2.5 getFirstNakNumber	2586
6.577.2.6 getLastNakNumber	2586
6.577.2.7 operator=	2586
6.577.2.8 setFirstNakNumber	2586

6.577.2.9	setLastNakNumber	2586
6.577.2.10	toString	2586
6.577.2.11	visit	2587
6.577.3	Field Documentation	2587
6.577.3.1	firstNakNumber	2587
6.577.3.2	ID_REPLAYCOMMAND	2587
6.577.3.3	lastNakNumber	2587
6.578	activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller Class	
	Reference	2587
6.578.1	Detailed Description	2588
6.578.2	Constructor & Destructor Documentation	2588
6.578.2.1	ReplayCommandMarshaller	2588
6.578.2.2	~ReplayCommandMarshaller	2588
6.578.3	Member Function Documentation	2588
6.578.3.1	createObject	2588
6.578.3.2	getDataStructureType	2589
6.578.3.3	looseMarshal	2589
6.578.3.4	looseUnmarshal	2589
6.578.3.5	tightMarshal1	2590
6.578.3.6	tightMarshal2	2590
6.578.3.7	tightUnmarshal	2590
6.579	activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller Class	
	Reference	2591
6.579.1	Detailed Description	2592
6.579.2	Constructor & Destructor Documentation	2592
6.579.2.1	ReplayCommandMarshaller	2592
6.579.2.2	~ReplayCommandMarshaller	2592
6.579.3	Member Function Documentation	2592
6.579.3.1	createObject	2592
6.579.3.2	getDataStructureType	2592
6.579.3.3	looseMarshal	2593
6.579.3.4	looseUnmarshal	2593
6.579.3.5	tightMarshal1	2593
6.579.3.6	tightMarshal2	2594
6.579.3.7	tightUnmarshal	2594
6.580	activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller Class	
	Reference	2595

6.580.1 Detailed Description	2596
6.580.2 Constructor & Destructor Documentation	2596
6.580.2.1 ReplayCommandMarshaller	2596
6.580.2.2 ~ReplayCommandMarshaller	2596
6.580.3 Member Function Documentation	2596
6.580.3.1 createObject	2596
6.580.3.2 getDataStructureType	2596
6.580.3.3 looseMarshal	2597
6.580.3.4 looseUnmarshal	2597
6.580.3.5 tightMarshal1	2597
6.580.3.6 tightMarshal2	2598
6.580.3.7 tightUnmarshal	2598
6.581activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller Class	
Reference	2599
6.581.1 Detailed Description	2600
6.581.2 Constructor & Destructor Documentation	2600
6.581.2.1 ReplayCommandMarshaller	2600
6.581.2.2 ~ReplayCommandMarshaller	2600
6.581.3 Member Function Documentation	2600
6.581.3.1 createObject	2600
6.581.3.2 getDataStructureType	2600
6.581.3.3 looseMarshal	2601
6.581.3.4 looseUnmarshal	2601
6.581.3.5 tightMarshal1	2601
6.581.3.6 tightMarshal2	2602
6.581.3.7 tightUnmarshal	2602
6.582activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller Class	
Reference	2603
6.582.1 Detailed Description	2604
6.582.2 Constructor & Destructor Documentation	2604
6.582.2.1 ReplayCommandMarshaller	2604
6.582.2.2 ~ReplayCommandMarshaller	2604
6.582.3 Member Function Documentation	2604
6.582.3.1 createObject	2604
6.582.3.2 getDataStructureType	2604
6.582.3.3 looseMarshal	2605
6.582.3.4 looseUnmarshal	2605

6.582.3.5 tightMarshal	2605
6.582.3.6 tightMarshal2	2606
6.582.3.7 tightUnmarshal	2606
6.583activemq::cmsutil::CmsTemplate::ResolveProducerExecutor Class Reference	2607
6.583.1 Constructor & Destructor Documentation	2607
6.583.1.1 ResolveProducerExecutor	2607
6.583.1.2 ~ResolveProducerExecutor	2607
6.583.2 Member Function Documentation	2607
6.583.2.1 getDestination	2607
6.584activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor Class Reference	2608
6.584.1 Constructor & Destructor Documentation	2608
6.584.1.1 ResolveReceiveExecutor	2608
6.584.1.2 ~ResolveReceiveExecutor	2608
6.584.2 Member Function Documentation	2608
6.584.2.1 getDestination	2608
6.585activemq::cmsutil::ResourceLifecycleManager Class Reference	2608
6.585.1 Detailed Description	2609
6.585.2 Constructor & Destructor Documentation	2609
6.585.2.1 ResourceLifecycleManager	2609
6.585.2.2 ~ResourceLifecycleManager	2609
6.585.3 Member Function Documentation	2609
6.585.3.1 addConnection	2609
6.585.3.2 addDestination	2610
6.585.3.3 addMessageConsumer	2610
6.585.3.4 addMessageProducer	2610
6.585.3.5 addSession	2610
6.585.3.6 destroy	2610
6.585.3.7 releaseAll	2611
6.586activemq::commands::Response Class Reference	2611
6.586.1 Constructor & Destructor Documentation	2612
6.586.1.1 Response	2612
6.586.1.2 Response	2612
6.586.1.3 ~Response	2612
6.586.2 Member Function Documentation	2612
6.586.2.1 cloneDataStructure	2612
6.586.2.2 copyDataStructure	2612

6.586.2.3 equals	2613
6.586.2.4 getCorrelationId	2613
6.586.2.5 getDataStructureType	2613
6.586.2.6 isResponse	2613
6.586.2.7 operator=	2614
6.586.2.8 setCorrelationId	2614
6.586.2.9 toString	2614
6.586.2.10 visit	2614
6.586.3 Field Documentation	2614
6.586.3.1 correlationId	2614
6.586.3.2 ID_RESPONSE	2614
6.587activemq::transport::mock::ResponseBuilder Class Reference	2614
6.587.1 Detailed Description	2615
6.587.2 Constructor & Destructor Documentation	2615
6.587.2.1 ~ResponseBuilder	2615
6.587.3 Member Function Documentation	2615
6.587.3.1 buildIncomingCommands	2615
6.587.3.2 buildResponse	2616
6.588activemq::transport::correlator::ResponseCorrelator Class Reference	2616
6.588.1 Detailed Description	2617
6.588.2 Constructor & Destructor Documentation	2617
6.588.2.1 ResponseCorrelator	2617
6.588.2.2 ~ResponseCorrelator	2617
6.588.3 Member Function Documentation	2617
6.588.3.1 close	2617
6.588.3.2 onCommand	2618
6.588.3.3 oneway	2618
6.588.3.4 onTransportException	2618
6.588.3.5 request	2618
6.588.3.6 request	2619
6.588.3.7 start	2619
6.589activemq::wireformat::openwire::marshal::v2::ResponseMarshaller Class Reference	2620
6.589.1 Detailed Description	2620
6.589.2 Constructor & Destructor Documentation	2621
6.589.2.1 ResponseMarshaller	2621
6.589.2.2 ~ResponseMarshaller	2621

6.589.3 Member Function Documentation	2621
6.589.3.1 createObject	2621
6.589.3.2 getDataStructureType	2621
6.589.3.3 looseMarshal	2621
6.589.3.4 looseUnmarshal	2622
6.589.3.5 tightMarshal1	2622
6.589.3.6 tightMarshal2	2623
6.589.3.7 tightUnmarshal	2623
6.590activemq::wireformat::openwire::marshal::v3::ResponseMarshaller Class Reference .	2624
6.590.1 Detailed Description	2625
6.590.2 Constructor & Destructor Documentation	2625
6.590.2.1 ResponseMarshaller	2625
6.590.2.2 ~ResponseMarshaller	2625
6.590.3 Member Function Documentation	2625
6.590.3.1 createObject	2625
6.590.3.2 getDataStructureType	2626
6.590.3.3 looseMarshal	2626
6.590.3.4 looseUnmarshal	2626
6.590.3.5 tightMarshal1	2627
6.590.3.6 tightMarshal2	2627
6.590.3.7 tightUnmarshal	2628
6.591activemq::wireformat::openwire::marshal::v5::ResponseMarshaller Class Reference .	2628
6.591.1 Detailed Description	2629
6.591.2 Constructor & Destructor Documentation	2630
6.591.2.1 ResponseMarshaller	2630
6.591.2.2 ~ResponseMarshaller	2630
6.591.3 Member Function Documentation	2630
6.591.3.1 createObject	2630
6.591.3.2 getDataStructureType	2630
6.591.3.3 looseMarshal	2630
6.591.3.4 looseUnmarshal	2631
6.591.3.5 tightMarshal1	2631
6.591.3.6 tightMarshal2	2632
6.591.3.7 tightUnmarshal	2632
6.592activemq::wireformat::openwire::marshal::v1::ResponseMarshaller Class Reference .	2633
6.592.1 Detailed Description	2634

6.592.2 Constructor & Destructor Documentation	2634
6.592.2.1 ResponseMarshaller	2634
6.592.2.2 ~ResponseMarshaller	2634
6.592.3 Member Function Documentation	2634
6.592.3.1 createObject	2634
6.592.3.2 getDataStructureType	2635
6.592.3.3 looseMarshal	2635
6.592.3.4 looseUnmarshal	2635
6.592.3.5 tightMarshal1	2636
6.592.3.6 tightMarshal2	2636
6.592.3.7 tightUnmarshal	2637
6.593activemq::wireformat::openwire::marshal::v4::ResponseMarshaller Class Reference	2637
6.593.1 Detailed Description	2638
6.593.2 Constructor & Destructor Documentation	2639
6.593.2.1 ResponseMarshaller	2639
6.593.2.2 ~ResponseMarshaller	2639
6.593.3 Member Function Documentation	2639
6.593.3.1 createObject	2639
6.593.3.2 getDataStructureType	2639
6.593.3.3 looseMarshal	2639
6.593.3.4 looseUnmarshal	2640
6.593.3.5 tightMarshal1	2640
6.593.3.6 tightMarshal2	2641
6.593.3.7 tightUnmarshal	2641
6.594decaf::lang::Runnable Class Reference	2642
6.594.1 Detailed Description	2642
6.594.2 Constructor & Destructor Documentation	2642
6.594.2.1 ~Runnable	2642
6.594.3 Member Function Documentation	2642
6.594.3.1 run	2642
6.595decaf::lang::Runtime Class Reference	2643
6.595.1 Constructor & Destructor Documentation	2643
6.595.1.1 ~Runtime	2643
6.595.2 Member Function Documentation	2643
6.595.2.1 getRuntime	2643
6.595.2.2 initializeRuntime	2644

6.595.2.3 initializeRuntime	2644
6.595.2.4 shutdownRuntime	2644
6.596decaf::lang::exceptions::RuntimeException Class Reference	2644
6.596.1 Constructor & Destructor Documentation	2645
6.596.1.1 RuntimeException	2645
6.596.1.2 RuntimeException	2645
6.596.1.3 RuntimeException	2645
6.596.1.4 RuntimeException	2646
6.596.1.5 RuntimeException	2646
6.596.1.6 RuntimeException	2646
6.596.1.7 ~RuntimeException	2646
6.596.2 Member Function Documentation	2646
6.596.2.1 clone	2646
6.597decaf::security_provider::SecurityProvider Class Reference	2647
6.597.1 Constructor & Destructor Documentation	2647
6.597.1.1 ~SecurityProvider	2647
6.597.2 Member Function Documentation	2647
6.597.2.1 createX500Principal	2647
6.597.2.2 createX500Principal	2647
6.597.2.3 createX500Principal	2647
6.598decaf::security_provider::SecurityProviderMap Class Reference	2648
6.598.1 Detailed Description	2648
6.598.2 Member Function Documentation	2648
6.598.2.1 getInstance	2648
6.598.2.2 getSecurityProviderNames	2648
6.598.2.3 lookup	2649
6.598.2.4 registerSecurityProvider	2649
6.598.2.5 unregisterSecurityProvider	2649
6.599decaf::security_provider::SecurityProviderRegistrar Class Reference	2649
6.599.1 Detailed Description	2650
6.599.2 Constructor & Destructor Documentation	2650
6.599.2.1 SecurityProviderRegistrar	2650
6.599.2.2 ~SecurityProviderRegistrar	2650
6.599.3 Member Function Documentation	2650
6.599.3.1 getProvider	2650
6.600decaf::util::concurrent::Semaphore Class Reference	2651

6.600.1 Detailed Description	2652
6.600.2 Constructor & Destructor Documentation	2653
6.600.2.1 Semaphore	2653
6.600.2.2 Semaphore	2654
6.600.2.3 ~Semaphore	2654
6.600.3 Member Function Documentation	2654
6.600.3.1 acquire	2654
6.600.3.2 acquire	2654
6.600.3.3 acquireUninterruptibly	2655
6.600.3.4 acquireUninterruptibly	2655
6.600.3.5 availablePermits	2656
6.600.3.6 drainPermits	2656
6.600.3.7 isFair	2656
6.600.3.8 release	2657
6.600.3.9 release	2657
6.600.3.10 oString	2657
6.600.3.11 tryAcquire	2658
6.600.3.12 tryAcquire	2658
6.600.3.13 tryAcquire	2659
6.600.3.14 tryAcquire	2660
6.601 activemq::cmsutil::CmsTemplate::SendExecutor Class Reference	2660
6.601.1 Constructor & Destructor Documentation	2661
6.601.1.1 SendExecutor	2661
6.601.1.2 ~SendExecutor	2661
6.601.2 Member Function Documentation	2661
6.601.2.1 doInCms	2661
6.602 decaf::net::ServerSocket Class Reference	2661
6.602.1 Detailed Description	2662
6.602.2 Member Typedef Documentation	2662
6.602.2.1 SocketAddress	2662
6.602.2.2 SocketHandle	2662
6.602.3 Constructor & Destructor Documentation	2662
6.602.3.1 ServerSocket	2662
6.602.3.2 ~ServerSocket	2662
6.602.4 Member Function Documentation	2662
6.602.4.1 accept	2662

6.602.4.2 bind	2663
6.602.4.3 bind	2663
6.602.4.4 close	2663
6.602.4.5 isBound	2663
6.603cms::Session Class Reference	2663
6.603.1 Detailed Description	2666
6.603.2 Member Enumeration Documentation	2667
6.603.2.1 AcknowledgeMode	2667
6.603.3 Constructor & Destructor Documentation	2667
6.603.3.1 ~Session	2667
6.603.4 Member Function Documentation	2667
6.603.4.1 close	2667
6.603.4.2 commit	2667
6.603.4.3 createBrowser	2668
6.603.4.4 createBrowser	2668
6.603.4.5 createBytesMessage	2669
6.603.4.6 createBytesMessage	2669
6.603.4.7 createConsumer	2669
6.603.4.8 createConsumer	2670
6.603.4.9 createConsumer	2670
6.603.4.10createDurableConsumer	2671
6.603.4.11createMapMessage	2671
6.603.4.12reateMessage	2671
6.603.4.13reateProducer	2672
6.603.4.14reateQueue	2672
6.603.4.15createStreamMessage	2672
6.603.4.16reateTemporaryQueue	2673
6.603.4.17reateTemporaryTopic	2673
6.603.4.18reateTextMessage	2673
6.603.4.19reateTextMessage	2674
6.603.4.20reateTopic	2674
6.603.4.21getAcknowledgeMode	2674
6.603.4.22sTransacted	2674
6.603.4.23recover	2675
6.603.4.24rollback	2675
6.603.4.25unsubscribe	2676

6.604	activemq::cmsutil::SessionCallback Class Reference	2676
6.604.1	Detailed Description	2676
6.604.2	Constructor & Destructor Documentation	2677
6.604.2.1	~SessionCallback	2677
6.604.3	Member Function Documentation	2677
6.604.3.1	doInCms	2677
6.605	activemq::commands::SessionId Class Reference	2677
6.605.1	Member Typedef Documentation	2679
6.605.1.1	COMPARATOR	2679
6.605.2	Constructor & Destructor Documentation	2679
6.605.2.1	SessionId	2679
6.605.2.2	SessionId	2679
6.605.2.3	SessionId	2679
6.605.2.4	SessionId	2679
6.605.2.5	SessionId	2679
6.605.2.6	~SessionId	2679
6.605.3	Member Function Documentation	2679
6.605.3.1	cloneDataStructure	2679
6.605.3.2	compareTo	2679
6.605.3.3	copyDataStructure	2679
6.605.3.4	equals	2679
6.605.3.5	equals	2680
6.605.3.6	getConnectionId	2680
6.605.3.7	getConnectionId	2680
6.605.3.8	getDataStructureType	2680
6.605.3.9	getParentId	2680
6.605.3.10	getValue	2680
6.605.3.11	operator<	2681
6.605.3.12	operator=	2681
6.605.3.13	operator==	2681
6.605.3.14	setConnectionId	2681
6.605.3.15	setValue	2681
6.605.3.16	toString	2681
6.605.4	Field Documentation	2681
6.605.4.1	connectionId	2681
6.605.4.2	ID_SESSIONID	2681

6.605.4.3 value	2681
6.606activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller Class Reference	2681
6.606.1 Detailed Description	2682
6.606.2 Constructor & Destructor Documentation	2683
6.606.2.1 SessionIdMarshaller	2683
6.606.2.2 ~SessionIdMarshaller	2683
6.606.3 Member Function Documentation	2683
6.606.3.1 createObject	2683
6.606.3.2 getDataStructureType	2683
6.606.3.3 looseMarshal	2683
6.606.3.4 looseUnmarshal	2684
6.606.3.5 tightMarshal1	2684
6.606.3.6 tightMarshal2	2684
6.606.3.7 tightUnmarshal	2685
6.607activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller Class Reference	2685
6.607.1 Detailed Description	2686
6.607.2 Constructor & Destructor Documentation	2687
6.607.2.1 SessionIdMarshaller	2687
6.607.2.2 ~SessionIdMarshaller	2687
6.607.3 Member Function Documentation	2687
6.607.3.1 createObject	2687
6.607.3.2 getDataStructureType	2687
6.607.3.3 looseMarshal	2687
6.607.3.4 looseUnmarshal	2688
6.607.3.5 tightMarshal1	2688
6.607.3.6 tightMarshal2	2688
6.607.3.7 tightUnmarshal	2689
6.608activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller Class Reference	2689
6.608.1 Detailed Description	2690
6.608.2 Constructor & Destructor Documentation	2691
6.608.2.1 SessionIdMarshaller	2691
6.608.2.2 ~SessionIdMarshaller	2691
6.608.3 Member Function Documentation	2691
6.608.3.1 createObject	2691
6.608.3.2 getDataStructureType	2691
6.608.3.3 looseMarshal	2691

6.608.3.4 looseUnmarshal	2692
6.608.3.5 tightMarshal1	2692
6.608.3.6 tightMarshal2	2692
6.608.3.7 tightUnmarshal	2693
6.609activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller Class Reference .	2693
6.609.1 Detailed Description	2694
6.609.2 Constructor & Destructor Documentation	2695
6.609.2.1 SessionIdMarshaller	2695
6.609.2.2 ~SessionIdMarshaller	2695
6.609.3 Member Function Documentation	2695
6.609.3.1 createObject	2695
6.609.3.2 getDataStructureType	2695
6.609.3.3 looseMarshal	2695
6.609.3.4 looseUnmarshal	2696
6.609.3.5 tightMarshal1	2696
6.609.3.6 tightMarshal2	2696
6.609.3.7 tightUnmarshal	2697
6.610activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller Class Reference .	2697
6.610.1 Detailed Description	2698
6.610.2 Constructor & Destructor Documentation	2699
6.610.2.1 SessionIdMarshaller	2699
6.610.2.2 ~SessionIdMarshaller	2699
6.610.3 Member Function Documentation	2699
6.610.3.1 createObject	2699
6.610.3.2 getDataStructureType	2699
6.610.3.3 looseMarshal	2699
6.610.3.4 looseUnmarshal	2700
6.610.3.5 tightMarshal1	2700
6.610.3.6 tightMarshal2	2700
6.610.3.7 tightUnmarshal	2701
6.611activemq::commands::SessionInfo Class Reference	2701
6.611.1 Constructor & Destructor Documentation	2703
6.611.1.1 SessionInfo	2703
6.611.1.2 SessionInfo	2703
6.611.1.3 ~SessionInfo	2703
6.611.2 Member Function Documentation	2703

6.611.2.1 cloneDataStructure	2703
6.611.2.2 copyDataStructure	2703
6.611.2.3 equals	2703
6.611.2.4 getAckMode	2704
6.611.2.5 getDataStructureType	2704
6.611.2.6 getSessionId	2704
6.611.2.7 getSessionId	2704
6.611.2.8 operator=	2704
6.611.2.9 setAckMode	2704
6.611.2.10 setSessionId	2704
6.611.2.11 toString	2704
6.611.2.12 visit	2704
6.611.3 Field Documentation	2705
6.611.3.1 ID_SESSIONINFO	2705
6.611.3.2 sessionId	2705
6.612activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller Class Reference	2705
6.612.1 Detailed Description	2706
6.612.2 Constructor & Destructor Documentation	2706
6.612.2.1 SessionInfoMarshaller	2706
6.612.2.2 ~SessionInfoMarshaller	2706
6.612.3 Member Function Documentation	2706
6.612.3.1 createObject	2706
6.612.3.2 getDataStructureType	2706
6.612.3.3 looseMarshal	2707
6.612.3.4 looseUnmarshal	2707
6.612.3.5 tightMarshal1	2707
6.612.3.6 tightMarshal2	2708
6.612.3.7 tightUnmarshal	2708
6.613activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller Class Reference	2709
6.613.1 Detailed Description	2710
6.613.2 Constructor & Destructor Documentation	2710
6.613.2.1 SessionInfoMarshaller	2710
6.613.2.2 ~SessionInfoMarshaller	2710
6.613.3 Member Function Documentation	2710
6.613.3.1 createObject	2710
6.613.3.2 getDataStructureType	2710

6.613.3.3 looseMarshal	2711
6.613.3.4 looseUnmarshal	2711
6.613.3.5 tightMarshal1	2711
6.613.3.6 tightMarshal2	2712
6.613.3.7 tightUnmarshal	2712
6.614activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller Class Reference	2713
6.614.1 Detailed Description	2714
6.614.2 Constructor & Destructor Documentation	2714
6.614.2.1 SessionInfoMarshaller	2714
6.614.2.2 ~SessionInfoMarshaller	2714
6.614.3 Member Function Documentation	2714
6.614.3.1 createObject	2714
6.614.3.2 getDataStructureType	2714
6.614.3.3 looseMarshal	2715
6.614.3.4 looseUnmarshal	2715
6.614.3.5 tightMarshal1	2715
6.614.3.6 tightMarshal2	2716
6.614.3.7 tightUnmarshal	2716
6.615activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller Class Reference	2717
6.615.1 Detailed Description	2718
6.615.2 Constructor & Destructor Documentation	2718
6.615.2.1 SessionInfoMarshaller	2718
6.615.2.2 ~SessionInfoMarshaller	2718
6.615.3 Member Function Documentation	2718
6.615.3.1 createObject	2718
6.615.3.2 getDataStructureType	2718
6.615.3.3 looseMarshal	2719
6.615.3.4 looseUnmarshal	2719
6.615.3.5 tightMarshal1	2719
6.615.3.6 tightMarshal2	2720
6.615.3.7 tightUnmarshal	2720
6.616activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller Class Reference	2721
6.616.1 Detailed Description	2722
6.616.2 Constructor & Destructor Documentation	2722
6.616.2.1 SessionInfoMarshaller	2722
6.616.2.2 ~SessionInfoMarshaller	2722

6.616.3 Member Function Documentation	2722
6.616.3.1 createObject	2722
6.616.3.2 getDataStructureType	2722
6.616.3.3 looseMarshal	2723
6.616.3.4 looseUnmarshal	2723
6.616.3.5 tightMarshal1	2723
6.616.3.6 tightMarshal2	2724
6.616.3.7 tightUnmarshal	2724
6.617activemq::cmsutil::SessionPool Class Reference	2725
6.617.1 Detailed Description	2725
6.617.2 Constructor & Destructor Documentation	2725
6.617.2.1 SessionPool	2725
6.617.2.2 ~SessionPool	2726
6.617.3 Member Function Documentation	2726
6.617.3.1 getResourceLifecycleManager	2726
6.617.3.2 returnSession	2726
6.617.3.3 takeSession	2726
6.618activemq::state::SessionState Class Reference	2726
6.618.1 Constructor & Destructor Documentation	2728
6.618.1.1 SessionState	2728
6.618.1.2 ~SessionState	2728
6.618.2 Member Function Documentation	2728
6.618.2.1 addConsumer	2728
6.618.2.2 addProducer	2728
6.618.2.3 checkShutdown	2728
6.618.2.4 getConsumerState	2728
6.618.2.5 getConsumerStates	2728
6.618.2.6 getInfo	2728
6.618.2.7 getProducerState	2728
6.618.2.8 getProducerStates	2728
6.618.2.9 removeConsumer	2728
6.618.2.10removeProducer	2728
6.618.2.11shutdown	2728
6.618.2.12toString	2728
6.619decaf::util::Set< E > Class Template Reference	2729
6.619.1 Detailed Description	2729

6.619.2 Constructor & Destructor Documentation	2729
6.619.2.1 ~Set	2729
6.620decaf::lang::Short Class Reference	2729
6.620.1 Constructor & Destructor Documentation	2731
6.620.1.1 Short	2731
6.620.1.2 Short	2732
6.620.1.3 ~Short	2732
6.620.2 Member Function Documentation	2732
6.620.2.1 byteValue	2732
6.620.2.2 compareTo	2732
6.620.2.3 compareTo	2732
6.620.2.4 decode	2733
6.620.2.5 doubleValue	2733
6.620.2.6 equals	2733
6.620.2.7 equals	2733
6.620.2.8 floatValue	2733
6.620.2.9 intValue	2734
6.620.2.10longValue	2734
6.620.2.11operator<	2734
6.620.2.12operator<	2734
6.620.2.13operator==	2735
6.620.2.14operator==	2735
6.620.2.15parseShort	2735
6.620.2.16parseShort	2736
6.620.2.17reverseBytes	2736
6.620.2.18shortValue	2736
6.620.2.19oString	2736
6.620.2.20oString	2737
6.620.2.21valueOf	2737
6.620.2.22valueOf	2737
6.620.2.23valueOf	2737
6.620.3 Field Documentation	2738
6.620.3.1 MAX_VALUE	2738
6.620.3.2 MIN_VALUE	2738
6.620.3.3 SIZE	2738
6.621decaf::internal::nio::ShortArrayBuffer Class Reference	2738

6.621.1 Constructor & Destructor Documentation	2740
6.621.1.1 ShortArrayBuffer	2740
6.621.1.2 ShortArrayBuffer	2740
6.621.1.3 ShortArrayBuffer	2740
6.621.1.4 ShortArrayBuffer	2741
6.621.1.5 ~ShortArrayBuffer	2741
6.621.2 Member Function Documentation	2741
6.621.2.1 array	2741
6.621.2.2 arrayOffset	2741
6.621.2.3 asReadOnlyBuffer	2742
6.621.2.4 compact	2742
6.621.2.5 duplicate	2743
6.621.2.6 get	2743
6.621.2.7 get	2743
6.621.2.8 hasArray	2744
6.621.2.9 isReadOnly	2744
6.621.2.10put	2744
6.621.2.11put	2744
6.621.2.12setReadOnly	2745
6.621.2.13slice	2745
6.622decaf::nio::ShortBuffer Class Reference	2745
6.622.1 Detailed Description	2747
6.622.2 Constructor & Destructor Documentation	2748
6.622.2.1 ShortBuffer	2748
6.622.2.2 ~ShortBuffer	2748
6.622.3 Member Function Documentation	2748
6.622.3.1 allocate	2748
6.622.3.2 array	2748
6.622.3.3 arrayOffset	2749
6.622.3.4 asReadOnlyBuffer	2749
6.622.3.5 compact	2750
6.622.3.6 compareTo	2750
6.622.3.7 duplicate	2750
6.622.3.8 equals	2751
6.622.3.9 get	2751
6.622.3.10get	2751

6.622.3.11	get	2751
6.622.3.12	get	2752
6.622.3.13	hasArray	2752
6.622.3.14	operator<	2753
6.622.3.15	operator==	2753
6.622.3.16	put	2753
6.622.3.17	put	2753
6.622.3.18	put	2754
6.622.3.19	put	2754
6.622.3.20	put	2755
6.622.3.21	slice	2755
6.622.3.22	toString	2756
6.622.3.23	wrap	2756
6.622.3.24	wrap	2756
6.623	activemq::commands::ShutdownInfo Class Reference	2757
6.623.1	Constructor & Destructor Documentation	2758
6.623.1.1	ShutdownInfo	2758
6.623.1.2	ShutdownInfo	2758
6.623.1.3	~ShutdownInfo	2758
6.623.2	Member Function Documentation	2758
6.623.2.1	cloneDataStructure	2758
6.623.2.2	copyDataStructure	2758
6.623.2.3	equals	2758
6.623.2.4	getDataStructureType	2758
6.623.2.5	isShutdownInfo	2759
6.623.2.6	operator=	2759
6.623.2.7	toString	2759
6.623.2.8	visit	2759
6.623.3	Field Documentation	2759
6.623.3.1	ID_SHUTDOWNINFO	2759
6.624	activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller Class Refer- ence	2760
6.624.1	Detailed Description	2760
6.624.2	Constructor & Destructor Documentation	2761
6.624.2.1	ShutdownInfoMarshaller	2761
6.624.2.2	~ShutdownInfoMarshaller	2761
6.624.3	Member Function Documentation	2761

6.624.3.1 createObject	2761
6.624.3.2 getDataStructureType	2761
6.624.3.3 looseMarshal	2761
6.624.3.4 looseUnmarshal	2762
6.624.3.5 tightMarshal1	2762
6.624.3.6 tightMarshal2	2762
6.624.3.7 tightUnmarshal	2763
6.625activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller Class Reference	2763
6.625.1 Detailed Description	2764
6.625.2 Constructor & Destructor Documentation	2765
6.625.2.1 ShutdownInfoMarshaller	2765
6.625.2.2 ~ShutdownInfoMarshaller	2765
6.625.3 Member Function Documentation	2765
6.625.3.1 createObject	2765
6.625.3.2 getDataStructureType	2765
6.625.3.3 looseMarshal	2765
6.625.3.4 looseUnmarshal	2766
6.625.3.5 tightMarshal1	2766
6.625.3.6 tightMarshal2	2766
6.625.3.7 tightUnmarshal	2767
6.626activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller Class Reference	2767
6.626.1 Detailed Description	2768
6.626.2 Constructor & Destructor Documentation	2769
6.626.2.1 ShutdownInfoMarshaller	2769
6.626.2.2 ~ShutdownInfoMarshaller	2769
6.626.3 Member Function Documentation	2769
6.626.3.1 createObject	2769
6.626.3.2 getDataStructureType	2769
6.626.3.3 looseMarshal	2769
6.626.3.4 looseUnmarshal	2770
6.626.3.5 tightMarshal1	2770
6.626.3.6 tightMarshal2	2770
6.626.3.7 tightUnmarshal	2771
6.627activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller Class Reference	2771

6.627.1 Detailed Description	2772
6.627.2 Constructor & Destructor Documentation	2773
6.627.2.1 ShutdownInfoMarshaller	2773
6.627.2.2 ~ShutdownInfoMarshaller	2773
6.627.3 Member Function Documentation	2773
6.627.3.1 createObject	2773
6.627.3.2 getDataStructureType	2773
6.627.3.3 looseMarshal	2773
6.627.3.4 looseUnmarshal	2774
6.627.3.5 tightMarshal1	2774
6.627.3.6 tightMarshal2	2774
6.627.3.7 tightUnmarshal	2775
6.628activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller Class Reference	2775
6.628.1 Detailed Description	2776
6.628.2 Constructor & Destructor Documentation	2777
6.628.2.1 ShutdownInfoMarshaller	2777
6.628.2.2 ~ShutdownInfoMarshaller	2777
6.628.3 Member Function Documentation	2777
6.628.3.1 createObject	2777
6.628.3.2 getDataStructureType	2777
6.628.3.3 looseMarshal	2777
6.628.3.4 looseUnmarshal	2778
6.628.3.5 tightMarshal1	2778
6.628.3.6 tightMarshal2	2778
6.628.3.7 tightUnmarshal	2779
6.629decaf::security::SignatureException Class Reference	2779
6.629.1 Constructor & Destructor Documentation	2780
6.629.1.1 SignatureException	2780
6.629.1.2 SignatureException	2780
6.629.1.3 SignatureException	2780
6.629.1.4 SignatureException	2781
6.629.1.5 SignatureException	2781
6.629.1.6 SignatureException	2781
6.629.1.7 ~SignatureException	2781
6.629.2 Member Function Documentation	2781
6.629.2.1 clone	2781

6.630decaf::util::logging::SimpleFormatter Class Reference	2782
6.630.1 Detailed Description	2782
6.630.2 Constructor & Destructor Documentation	2783
6.630.2.1 SimpleFormatter	2783
6.630.2.2 ~SimpleFormatter	2783
6.630.3 Member Function Documentation	2783
6.630.3.1 format	2783
6.630.3.2 formatMessage	2783
6.630.3.3 getHead	2783
6.630.3.4 getTail	2783
6.631decaf::util::logging::SimpleLogger Class Reference	2784
6.631.1 Constructor & Destructor Documentation	2785
6.631.1.1 SimpleLogger	2785
6.631.1.2 ~SimpleLogger	2785
6.631.2 Member Function Documentation	2785
6.631.2.1 debug	2785
6.631.2.2 error	2785
6.631.2.3 fatal	2785
6.631.2.4 info	2785
6.631.2.5 log	2785
6.631.2.6 mark	2785
6.631.2.7 warn	2785
6.632decaf::net::Socket Class Reference	2786
6.632.1 Member Typedef Documentation	2787
6.632.1.1 SocketAddress	2787
6.632.1.2 SocketHandle	2787
6.632.2 Constructor & Destructor Documentation	2787
6.632.2.1 ~Socket	2787
6.632.3 Member Function Documentation	2787
6.632.3.1 connect	2787
6.632.3.2 getInputStream	2788
6.632.3.3 getKeepAlive	2788
6.632.3.4 getOutputStream	2788
6.632.3.5 getReceiveBufferSize	2788
6.632.3.6 getReuseAddress	2789
6.632.3.7 getSendBufferSize	2789

6.632.3.8	getSoLinger	2789
6.632.3.9	getSoTimeout	2790
6.632.3.10	isConnected	2790
6.632.3.11	setKeepAlive	2790
6.632.3.12	setReceiveBufferSize	2790
6.632.3.13	setReuseAddress	2791
6.632.3.14	setSendBufferSize	2791
6.632.3.15	setSoLinger	2791
6.632.3.16	setSoTimeout	2792
6.633	decaf::net::SocketError Class Reference	2792
6.633.1	Detailed Description	2792
6.633.2	Member Function Documentation	2792
6.633.2.1	getErrorCode	2792
6.633.2.2	getErrorString	2792
6.634	decaf::net::SocketException Class Reference	2793
6.634.1	Detailed Description	2793
6.634.2	Constructor & Destructor Documentation	2793
6.634.2.1	SocketException	2793
6.634.2.2	SocketException	2793
6.634.2.3	SocketException	2793
6.634.2.4	SocketException	2793
6.634.2.5	SocketException	2794
6.634.2.6	SocketException	2794
6.634.2.7	~SocketException	2794
6.634.3	Member Function Documentation	2794
6.634.3.1	clone	2794
6.635	decaf::net::SocketFactory Class Reference	2795
6.635.1	Detailed Description	2795
6.635.2	Constructor & Destructor Documentation	2796
6.635.2.1	~SocketFactory	2796
6.635.3	Member Function Documentation	2796
6.635.3.1	createSocket	2796
6.636	decaf::net::SocketInputStream Class Reference	2796
6.636.1	Detailed Description	2798
6.636.2	Constructor & Destructor Documentation	2798
6.636.2.1	SocketInputStream	2798

6.636.2.2 ~SocketInputStream	2798
6.636.3 Member Function Documentation	2798
6.636.3.1 available	2798
6.636.3.2 close	2798
6.636.3.3 lock	2799
6.636.3.4 mark	2799
6.636.3.5 markSupported	2799
6.636.3.6 notify	2799
6.636.3.7 notifyAll	2800
6.636.3.8 read	2800
6.636.3.9 read	2800
6.636.3.10 reset	2801
6.636.3.11 skip	2801
6.636.3.12 tryLock	2801
6.636.3.13 unlock	2802
6.636.3.14 wait	2802
6.636.3.15 wait	2802
6.636.3.16 wait	2803
6.637 decaf::net::SocketOutputStream Class Reference	2803
6.637.1 Detailed Description	2804
6.637.2 Constructor & Destructor Documentation	2805
6.637.2.1 SocketOutputStream	2805
6.637.2.2 ~SocketOutputStream	2805
6.637.3 Member Function Documentation	2805
6.637.3.1 close	2805
6.637.3.2 flush	2805
6.637.3.3 lock	2805
6.637.3.4 notify	2806
6.637.3.5 notifyAll	2806
6.637.3.6 tryLock	2806
6.637.3.7 unlock	2806
6.637.3.8 wait	2807
6.637.3.9 wait	2807
6.637.3.10 wait	2807
6.637.3.11 write	2808
6.637.3.12 write	2808

6.637.3.13	write	2809
6.638	decaf::net::SocketTimeoutException Class Reference	2809
6.638.1	Constructor & Destructor Documentation	2810
6.638.1.1	SocketTimeoutException	2810
6.638.1.2	SocketTimeoutException	2810
6.638.1.3	SocketTimeoutException	2810
6.638.1.4	SocketTimeoutException	2810
6.638.1.5	SocketTimeoutException	2811
6.638.1.6	SocketTimeoutException	2811
6.638.1.7	~SocketTimeoutException	2811
6.638.2	Member Function Documentation	2811
6.638.2.1	clone	2811
6.639	activemq::commands::BrokerError::StackTraceElement Struct Reference	2811
6.639.1	Field Documentation	2812
6.639.1.1	ClassName	2812
6.639.1.2	FileName	2812
6.639.1.3	LineNumber	2812
6.639.1.4	MethodName	2812
6.640	decaf::internal::io::StandardErrorOutputStream Class Reference	2812
6.640.1	Detailed Description	2813
6.640.2	Constructor & Destructor Documentation	2814
6.640.2.1	StandardErrorOutputStream	2814
6.640.2.2	~StandardErrorOutputStream	2814
6.640.3	Member Function Documentation	2814
6.640.3.1	close	2814
6.640.3.2	flush	2814
6.640.3.3	lock	2814
6.640.3.4	notify	2814
6.640.3.5	notifyAll	2815
6.640.3.6	tryLock	2815
6.640.3.7	unlock	2815
6.640.3.8	wait	2815
6.640.3.9	wait	2816
6.640.3.10	wait	2816
6.640.3.11	write	2817
6.640.3.12	write	2817

6.640.3.13	write	2817
6.641	decaf::internal::io::StandardInputStream Class Reference	2818
6.641.1	Constructor & Destructor Documentation	2820
6.641.1.1	StandardInputStream	2820
6.641.1.2	~StandardInputStream	2820
6.641.2	Member Function Documentation	2820
6.641.2.1	available	2820
6.641.2.2	close	2820
6.641.2.3	lock	2820
6.641.2.4	mark	2820
6.641.2.5	markSupported	2821
6.641.2.6	notify	2821
6.641.2.7	notifyAll	2821
6.641.2.8	read	2821
6.641.2.9	read	2822
6.641.2.10	reset	2822
6.641.2.11	skip	2823
6.641.2.12	tryLock	2823
6.641.2.13	unlock	2823
6.641.2.14	wait	2824
6.641.2.15	wait	2824
6.641.2.16	wait	2825
6.642	decaf::internal::io::StandardOutputStream Class Reference	2825
6.642.1	Constructor & Destructor Documentation	2826
6.642.1.1	StandardOutputStream	2826
6.642.1.2	~StandardOutputStream	2826
6.642.2	Member Function Documentation	2826
6.642.2.1	close	2826
6.642.2.2	flush	2827
6.642.2.3	lock	2827
6.642.2.4	notify	2827
6.642.2.5	notifyAll	2827
6.642.2.6	tryLock	2828
6.642.2.7	unlock	2828
6.642.2.8	wait	2828
6.642.2.9	wait	2828

6.642.2.10wait	2829
6.642.2.11write	2829
6.642.2.12write	2830
6.642.2.13write	2830
6.643cms::Startable Class Reference	2831
6.643.1 Detailed Description	2831
6.643.2 Constructor & Destructor Documentation	2831
6.643.2.1 ~Startable	2831
6.643.3 Member Function Documentation	2831
6.643.3.1 start	2831
6.644decaf::lang::STATIC_CAST_TOKEN Struct Reference	2831
6.645activemq::core::ActiveMQConstants::StaticInitializer Class Reference	2832
6.645.1 Constructor & Destructor Documentation	2832
6.645.1.1 StaticInitializer	2832
6.645.1.2 ~StaticInitializer	2832
6.645.2 Field Documentation	2832
6.645.2.1 destOptionMap	2832
6.645.2.2 destOptions	2832
6.645.2.3 uriParams	2832
6.645.2.4 uriParamsMap	2832
6.646decaf::util::StlList< E > Class Template Reference	2833
6.646.1 Detailed Description	2837
6.646.2 Constructor & Destructor Documentation	2837
6.646.2.1 StlList	2837
6.646.2.2 StlList	2838
6.646.2.3 StlList	2838
6.646.2.4 ~StlList	2838
6.646.3 Member Function Documentation	2838
6.646.3.1 add	2838
6.646.3.2 add	2839
6.646.3.3 addAll	2839
6.646.3.4 clear	2840
6.646.3.5 contains	2840
6.646.3.6 copy	2841
6.646.3.7 equals	2841
6.646.3.8 get	2841

6.646.3.9	indexOf	2841
6.646.3.10	isEmpty	2841
6.646.3.11	iterator	2842
6.646.3.12	iterator	2842
6.646.3.13	lastIndexOf	2842
6.646.3.14	listIterator	2842
6.646.3.15	listIterator	2843
6.646.3.16	listIterator	2843
6.646.3.17	listIterator	2843
6.646.3.18	remove	2843
6.646.3.19	remove	2843
6.646.3.20	set	2844
6.646.3.21	size	2844
6.647	decaf::util::StlMap< K, V, COMPARATOR > Class Template Reference	2845
6.647.1	Detailed Description	2849
6.647.2	Constructor & Destructor Documentation	2849
6.647.2.1	StlMap	2849
6.647.2.2	StlMap	2849
6.647.2.3	StlMap	2849
6.647.2.4	~StlMap	2850
6.647.3	Member Function Documentation	2850
6.647.3.1	clear	2850
6.647.3.2	containsKey	2850
6.647.3.3	containsValue	2850
6.647.3.4	copy	2851
6.647.3.5	copy	2851
6.647.3.6	equals	2851
6.647.3.7	equals	2851
6.647.3.8	get	2851
6.647.3.9	get	2852
6.647.3.10	isEmpty	2852
6.647.3.11	keySet	2852
6.647.3.12	lock	2853
6.647.3.13	notify	2853
6.647.3.14	notifyAll	2853
6.647.3.15	put	2854

6.647.3.16putAll	2854
6.647.3.17putAll	2854
6.647.3.18remove	2855
6.647.3.19size	2855
6.647.3.20tryLock	2855
6.647.3.21unlock	2856
6.647.3.22values	2856
6.647.3.23wait	2856
6.647.3.24wait	2856
6.647.3.25wait	2857
6.648decaf::util::StlQueue< T > Class Template Reference	2858
6.648.1 Detailed Description	2859
6.648.2 Constructor & Destructor Documentation	2860
6.648.2.1 StlQueue	2860
6.648.2.2 ~StlQueue	2860
6.648.3 Member Function Documentation	2860
6.648.3.1 back	2860
6.648.3.2 back	2860
6.648.3.3 clear	2860
6.648.3.4 empty	2861
6.648.3.5 enqueueFront	2861
6.648.3.6 front	2861
6.648.3.7 front	2861
6.648.3.8 getSafeValue	2861
6.648.3.9 iterator	2862
6.648.3.10lock	2862
6.648.3.11notify	2862
6.648.3.12notifyAll	2862
6.648.3.13pop	2863
6.648.3.14push	2863
6.648.3.15reverse	2863
6.648.3.16size	2863
6.648.3.17toArray	2863
6.648.3.18tryLock	2863
6.648.3.19unlock	2864
6.648.3.20wait	2864

6.648.3.21wait	2864
6.648.3.22wait	2865
6.649decaf::util::StlSet< E > Class Template Reference	2865
6.649.1 Detailed Description	2868
6.649.2 Constructor & Destructor Documentation	2868
6.649.2.1 StlSet	2868
6.649.2.2 StlSet	2868
6.649.2.3 StlSet	2868
6.649.2.4 ~StlSet	2868
6.649.3 Member Function Documentation	2868
6.649.3.1 add	2868
6.649.3.2 clear	2869
6.649.3.3 contains	2869
6.649.3.4 copy	2870
6.649.3.5 equals	2870
6.649.3.6 isEmpty	2870
6.649.3.7 iterator	2870
6.649.3.8 iterator	2870
6.649.3.9 remove	2870
6.649.3.10size	2871
6.650activemq::wireformat::stomp::StompCommandConstants Class Reference	2871
6.650.1 Field Documentation	2873
6.650.1.1 ABORT	2873
6.650.1.2 ACK	2873
6.650.1.3 ACK_AUTO	2873
6.650.1.4 ACK_CLIENT	2873
6.650.1.5 ACK_INDIVIDUAL	2873
6.650.1.6 BEGIN	2873
6.650.1.7 BYTES	2873
6.650.1.8 COMMIT	2873
6.650.1.9 CONNECT	2873
6.650.1.10CONNECTED	2873
6.650.1.11DISCONNECT	2873
6.650.1.12ERROR_CMD	2873
6.650.1.13HEADER_ACK	2873
6.650.1.14HEADER_CLIENT_ID	2873

6.650.1.15	HEADER_CONSUMERPRIORITY	2873
6.650.1.16	HEADER_CONTENTLENGTH	2873
6.650.1.17	HEADER_CORRELATIONID	2873
6.650.1.18	HEADER_DESTINATION	2873
6.650.1.19	HEADER_DISPATCH_ASYNC	2873
6.650.1.20	HEADER_EXCLUSIVE	2873
6.650.1.21	HEADER_EXPIRES	2873
6.650.1.22	HEADER_ID	2873
6.650.1.23	HEADER_JMSPRIORITY	2873
6.650.1.24	HEADER_LOGIN	2873
6.650.1.25	HEADER_MAXPENDINGMSGLIMIT	2873
6.650.1.26	HEADER_MESSAGE	2873
6.650.1.27	HEADER_MESSAGEID	2873
6.650.1.28	HEADER_NOLOCAL	2873
6.650.1.29	HEADER_OLDSUBSCRIPTIONNAME	2873
6.650.1.30	HEADER_PASSWORD	2873
6.650.1.31	HEADER_PERSISTENT	2873
6.650.1.32	HEADER_PREFETCHSIZE	2873
6.650.1.33	HEADER_RECEIPT_REQUIRED	2873
6.650.1.34	HEADER_RECEIPTID	2873
6.650.1.35	HEADER_REDELIVERED	2873
6.650.1.36	HEADER_REDELIVERYCOUNT	2873
6.650.1.37	HEADER_REPLYTO	2873
6.650.1.38	HEADER_REQUESTID	2873
6.650.1.39	HEADER_RESPONSEID	2873
6.650.1.40	HEADER_RETROACTIVE	2873
6.650.1.41	HEADER_SELECTOR	2873
6.650.1.42	HEADER_SESSIONID	2873
6.650.1.43	HEADER_SUBSCRIPTION	2873
6.650.1.44	HEADER_SUBSCRIPTIONNAME	2873
6.650.1.45	HEADER_TIMESTAMP	2873
6.650.1.46	HEADER_TRANSACTIONID	2873
6.650.1.47	HEADER_TRANSFORMATION	2873
6.650.1.48	HEADER_TRANSFORMATION_ERROR	2873
6.650.1.49	HEADER_TYPE	2873
6.650.1.50	MESSAGE	2873

6.650.1.51	QUEUE_PREFIX	2873
6.650.1.52	RECEIPT	2873
6.650.1.53	SEND	2873
6.650.1.54	SUBSCRIBE	2873
6.650.1.55	TEMPQUEUE_PREFIX	2873
6.650.1.56	TEMPTOPIC_PREFIX	2873
6.650.1.57	TEXT	2873
6.650.1.58	TOPIC_PREFIX	2873
6.650.1.59	UNSUBSCRIBE	2873
6.651	activemq::wireformat::stomp::StompFrame Class Reference	2874
6.651.1	Detailed Description	2875
6.651.2	Constructor & Destructor Documentation	2875
6.651.2.1	StompFrame	2875
6.651.2.2	~StompFrame	2875
6.651.3	Member Function Documentation	2875
6.651.3.1	clone	2875
6.651.3.2	copy	2875
6.651.3.3	fromStream	2876
6.651.3.4	getBody	2876
6.651.3.5	getBody	2876
6.651.3.6	getBodyLength	2876
6.651.3.7	getCommand	2876
6.651.3.8	getProperties	2877
6.651.3.9	getProperties	2877
6.651.3.10	getProperty	2877
6.651.3.11	hasProperty	2877
6.651.3.12	removeProperty	2877
6.651.3.13	setBody	2878
6.651.3.14	setCommand	2878
6.651.3.15	setProperty	2878
6.651.3.16	oStream	2878
6.652	activemq::wireformat::stomp::StompHelper Class Reference	2878
6.652.1	Detailed Description	2879
6.652.2	Constructor & Destructor Documentation	2880
6.652.2.1	StompHelper	2880
6.652.2.2	~StompHelper	2880

6.652.3 Member Function Documentation	2880
6.652.3.1 convertConsumerId	2880
6.652.3.2 convertConsumerId	2880
6.652.3.3 convertDestination	2880
6.652.3.4 convertDestination	2881
6.652.3.5 convertMessageId	2881
6.652.3.6 convertMessageId	2881
6.652.3.7 convertProducerId	2881
6.652.3.8 convertProducerId	2882
6.652.3.9 convertProperties	2882
6.652.3.10 convertProperties	2882
6.652.3.11 convertTransactionId	2882
6.652.3.12 convertTransactionId	2883
6.653 activemq::wireformat::stomp::StompWireFormat Class Reference	2883
6.653.1 Constructor & Destructor Documentation	2884
6.653.1.1 StompWireFormat	2884
6.653.1.2 ~StompWireFormat	2884
6.653.2 Member Function Documentation	2884
6.653.2.1 createNegotiator	2884
6.653.2.2 getVersion	2884
6.653.2.3 hasNegotiator	2885
6.653.2.4 inReceive	2885
6.653.2.5 marshal	2885
6.653.2.6 setVersion	2885
6.653.2.7 unmarshal	2886
6.654 activemq::wireformat::stomp::StompWireFormatFactory Class Reference	2886
6.654.1 Detailed Description	2886
6.654.2 Constructor & Destructor Documentation	2887
6.654.2.1 StompWireFormatFactory	2887
6.654.2.2 ~StompWireFormatFactory	2887
6.654.3 Member Function Documentation	2887
6.654.3.1 createWireFormat	2887
6.655 cms::Stoppable Class Reference	2887
6.655.1 Detailed Description	2887
6.655.2 Constructor & Destructor Documentation	2888
6.655.2.1 ~Stoppable	2888

6.655.3 Member Function Documentation	2888
6.655.3.1 stop	2888
6.656decaf::util::logging::StreamHandler Class Reference	2888
6.656.1 Constructor & Destructor Documentation	2889
6.656.1.1 StreamHandler	2889
6.656.1.2 StreamHandler	2889
6.656.1.3 ~StreamHandler	2889
6.656.2 Member Function Documentation	2889
6.656.2.1 close	2889
6.656.2.2 flush	2890
6.656.2.3 getFilter	2890
6.656.2.4 getFormatter	2890
6.656.2.5 getLevel	2890
6.656.2.6 getOutputStream	2890
6.656.2.7 isLoggable	2891
6.656.2.8 publish	2891
6.656.2.9 setFilter	2891
6.656.2.10setFormatter	2891
6.656.2.11setLevel	2892
6.657cms::StreamMessage Class Reference	2892
6.657.1 Detailed Description	2894
6.657.2 Constructor & Destructor Documentation	2895
6.657.2.1 ~StreamMessage	2895
6.657.3 Member Function Documentation	2895
6.657.3.1 readBoolean	2895
6.657.3.2 readByte	2895
6.657.3.3 readBytes	2896
6.657.3.4 readBytes	2896
6.657.3.5 readChar	2897
6.657.3.6 readDouble	2897
6.657.3.7 readFloat	2898
6.657.3.8 readInt	2898
6.657.3.9 readLong	2899
6.657.3.10readShort	2899
6.657.3.11readString	2899
6.657.3.12readUnsignedShort	2900

6.657.3.13	writeBoolean	2900
6.657.3.14	writeByte	2900
6.657.3.15	writeBytes	2901
6.657.3.16	writeBytes	2901
6.657.3.17	writeChar	2901
6.657.3.18	writeDouble	2902
6.657.3.19	writeFloat	2902
6.657.3.20	writeInt	2902
6.657.3.21	writeLong	2903
6.657.3.22	writeShort	2903
6.657.3.23	writeString	2903
6.657.3.24	writeUnsignedShort	2904
6.658	decaf::util::StringTokenizer Class Reference	2904
6.658.1	Constructor & Destructor Documentation	2905
6.658.1.1	StringTokenizer	2905
6.658.1.2	~StringTokenizer	2905
6.658.2	Member Function Documentation	2905
6.658.2.1	countTokens	2905
6.658.2.2	hasMoreTokens	2905
6.658.2.3	nextToken	2906
6.658.2.4	nextToken	2906
6.658.2.5	reset	2906
6.658.2.6	toArray	2907
6.659	activemq::commands::SubscriptionInfo Class Reference	2907
6.659.1	Constructor & Destructor Documentation	2908
6.659.1.1	SubscriptionInfo	2908
6.659.1.2	SubscriptionInfo	2908
6.659.1.3	~SubscriptionInfo	2908
6.659.2	Member Function Documentation	2908
6.659.2.1	cloneDataStructure	2908
6.659.2.2	copyDataStructure	2909
6.659.2.3	equals	2909
6.659.2.4	getClientId	2909
6.659.2.5	getClientId	2909
6.659.2.6	getDataStructureType	2909
6.659.2.7	getDestination	2910

6.659.2.8	getDestination	2910
6.659.2.9	getSelector	2910
6.659.2.10	getSelector	2910
6.659.2.11	getSubscriptionName	2910
6.659.2.12	getSubscriptionName	2910
6.659.2.13	getSubscribedDestination	2910
6.659.2.14	getSubscribedDestination	2910
6.659.2.15	operator=	2910
6.659.2.16	setClientId	2910
6.659.2.17	setDestination	2910
6.659.2.18	setSelector	2910
6.659.2.19	setSubscriptionName	2910
6.659.2.20	setSubscribedDestination	2910
6.659.2.21	toString	2910
6.659.3	Field Documentation	2911
6.659.3.1	clientId	2911
6.659.3.2	destination	2911
6.659.3.3	ID_SUBSCRIPTIONINFO	2911
6.659.3.4	selector	2911
6.659.3.5	subscriptionName	2911
6.659.3.6	subscribedDestination	2911
6.660	activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller Class	
	Reference	2911
6.660.1	Detailed Description	2912
6.660.2	Constructor & Destructor Documentation	2912
6.660.2.1	SubscriptionInfoMarshaller	2912
6.660.2.2	~SubscriptionInfoMarshaller	2912
6.660.3	Member Function Documentation	2912
6.660.3.1	createObject	2912
6.660.3.2	getDataStructureType	2913
6.660.3.3	looseMarshal	2913
6.660.3.4	looseUnmarshal	2913
6.660.3.5	tightMarshal1	2914
6.660.3.6	tightMarshal2	2914
6.660.3.7	tightUnmarshal	2915
6.661	activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller Class	
	Reference	2915

6.661.1 Detailed Description	2916
6.661.2 Constructor & Destructor Documentation	2916
6.661.2.1 SubscriptionInfoMarshaller	2916
6.661.2.2 ~SubscriptionInfoMarshaller	2916
6.661.3 Member Function Documentation	2916
6.661.3.1 createObject	2916
6.661.3.2 getDataStructureType	2917
6.661.3.3 looseMarshal	2917
6.661.3.4 looseUnmarshal	2917
6.661.3.5 tightMarshal1	2918
6.661.3.6 tightMarshal2	2918
6.661.3.7 tightUnmarshal	2918
6.662activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller Class	
Reference	2919
6.662.1 Detailed Description	2920
6.662.2 Constructor & Destructor Documentation	2920
6.662.2.1 SubscriptionInfoMarshaller	2920
6.662.2.2 ~SubscriptionInfoMarshaller	2920
6.662.3 Member Function Documentation	2920
6.662.3.1 createObject	2920
6.662.3.2 getDataStructureType	2920
6.662.3.3 looseMarshal	2921
6.662.3.4 looseUnmarshal	2921
6.662.3.5 tightMarshal1	2921
6.662.3.6 tightMarshal2	2922
6.662.3.7 tightUnmarshal	2922
6.663activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller Class	
Reference	2923
6.663.1 Detailed Description	2924
6.663.2 Constructor & Destructor Documentation	2924
6.663.2.1 SubscriptionInfoMarshaller	2924
6.663.2.2 ~SubscriptionInfoMarshaller	2924
6.663.3 Member Function Documentation	2924
6.663.3.1 createObject	2924
6.663.3.2 getDataStructureType	2924
6.663.3.3 looseMarshal	2924
6.663.3.4 looseUnmarshal	2925

6.663.3.5	tightMarshal1	2925
6.663.3.6	tightMarshal2	2926
6.663.3.7	tightUnmarshal	2926
6.664	activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller Class Reference	2926
6.664.1	Detailed Description	2927
6.664.2	Constructor & Destructor Documentation	2928
6.664.2.1	SubscriptionInfoMarshaller	2928
6.664.2.2	~SubscriptionInfoMarshaller	2928
6.664.3	Member Function Documentation	2928
6.664.3.1	createObject	2928
6.664.3.2	getDataStructureType	2928
6.664.3.3	looseMarshal	2928
6.664.3.4	looseUnmarshal	2929
6.664.3.5	tightMarshal1	2929
6.664.3.6	tightMarshal2	2929
6.664.3.7	tightUnmarshal	2930
6.665	decaf::util::concurrent::Synchronizable Class Reference	2930
6.665.1	Detailed Description	2931
6.665.2	Constructor & Destructor Documentation	2932
6.665.2.1	~Synchronizable	2932
6.665.3	Member Function Documentation	2932
6.665.3.1	lock	2932
6.665.3.2	notify	2933
6.665.3.3	notifyAll	2934
6.665.3.4	tryLock	2935
6.665.3.5	unlock	2936
6.665.3.6	wait	2937
6.665.3.7	wait	2938
6.665.3.8	wait	2940
6.666	decaf::internal::util::concurrent::SynchronizableImpl Class Reference	2941
6.666.1	Detailed Description	2942
6.666.2	Constructor & Destructor Documentation	2943
6.666.2.1	SynchronizableImpl	2943
6.666.2.2	~SynchronizableImpl	2943
6.666.3	Member Function Documentation	2943
6.666.3.1	lock	2943

6.666.3.2	notify	2943
6.666.3.3	notifyAll	2943
6.666.3.4	tryLock	2944
6.666.3.5	unlock	2944
6.666.3.6	wait	2944
6.666.3.7	wait	2944
6.666.3.8	wait	2945
6.667	activemq::core::Synchronization Class Reference	2945
6.667.1	Detailed Description	2946
6.667.2	Constructor & Destructor Documentation	2946
6.667.2.1	~Synchronization	2946
6.667.3	Member Function Documentation	2946
6.667.3.1	afterCommit	2946
6.667.3.2	afterRollback	2946
6.667.3.3	beforeEnd	2946
6.668	decaf::util::concurrent::SynchronousQueue< E > Class Template Reference	2946
6.668.1	Detailed Description	2948
6.668.2	Constructor & Destructor Documentation	2949
6.668.2.1	SynchronousQueue	2949
6.668.2.2	~SynchronousQueue	2949
6.668.3	Member Function Documentation	2949
6.668.3.1	clear	2949
6.668.3.2	contains	2949
6.668.3.3	containsAll	2949
6.668.3.4	drainTo	2949
6.668.3.5	drainTo	2949
6.668.3.6	equals	2950
6.668.3.7	isEmpty	2950
6.668.3.8	iterator	2950
6.668.3.9	iterator	2950
6.668.3.10	offer	2950
6.668.3.11	offer	2950
6.668.3.12	peek	2951
6.668.3.13	poll	2951
6.668.3.14	poll	2951
6.668.3.15	put	2952

6.668.3.16	remainingCapacity	2952
6.668.3.17	remove	2952
6.668.3.18	removeAll	2952
6.668.3.19	retainAll	2952
6.668.3.20	size	2952
6.668.3.21	take	2952
6.668.3.22	toArray	2953
6.669	decaf::lang::System Class Reference	2953
6.669.1	Constructor & Destructor Documentation	2954
6.669.1.1	System	2954
6.669.1.2	~System	2954
6.669.2	Member Function Documentation	2954
6.669.2.1	availableProcessors	2954
6.669.2.2	currentTimeMillis	2954
6.669.2.3	getenv	2954
6.669.2.4	getenv	2955
6.669.2.5	nanoTime	2955
6.669.2.6	setenv	2955
6.669.2.7	unsetenv	2956
6.670	activemq::threads::Task Class Reference	2956
6.670.1	Detailed Description	2956
6.670.2	Constructor & Destructor Documentation	2956
6.670.2.1	~Task	2956
6.670.3	Member Function Documentation	2956
6.670.3.1	iterate	2956
6.671	decaf::util::concurrent::TaskListener Class Reference	2957
6.671.1	Constructor & Destructor Documentation	2957
6.671.1.1	~TaskListener	2957
6.671.2	Member Function Documentation	2957
6.671.2.1	onTaskComplete	2957
6.671.2.2	onTaskException	2957
6.672	activemq::threads::TaskRunner Class Reference	2958
6.672.1	Constructor & Destructor Documentation	2958
6.672.1.1	~TaskRunner	2958
6.672.2	Member Function Documentation	2958
6.672.2.1	shutdown	2958

6.672.2.2 shutdown	2959
6.672.2.3 wakeup	2959
6.673decaf::net::TcpSocket Class Reference	2959
6.673.1 Detailed Description	2961
6.673.2 Constructor & Destructor Documentation	2961
6.673.2.1 TcpSocket	2961
6.673.2.2 TcpSocket	2961
6.673.2.3 ~TcpSocket	2961
6.673.3 Member Function Documentation	2961
6.673.3.1 checkResult	2961
6.673.3.2 close	2961
6.673.3.3 connect	2962
6.673.3.4 connect	2962
6.673.3.5 getInputStream	2962
6.673.3.6 getKeepAlive	2962
6.673.3.7 getOutputStream	2963
6.673.3.8 getReceiveBufferSize	2963
6.673.3.9 getReuseAddress	2963
6.673.3.10getSendBufferSize	2963
6.673.3.11getSocketHandle	2964
6.673.3.12getSoLinger	2964
6.673.3.13getSoTimeout	2964
6.673.3.14getTcpNoDelay	2964
6.673.3.15sConnected	2965
6.673.3.16setKeepAlive	2965
6.673.3.17setReceiveBufferSize	2965
6.673.3.18setReuseAddress	2965
6.673.3.19setSendBufferSize	2966
6.673.3.20setSoLinger	2966
6.673.3.21setSoTimeout	2966
6.673.3.22setTcpNoDelay	2966
6.674activemq::transport::tcp::TcpTransport Class Reference	2967
6.674.1 Detailed Description	2967
6.674.2 Constructor & Destructor Documentation	2968
6.674.2.1 TcpTransport	2968
6.674.2.2 TcpTransport	2968

6.674.2.3 ~TcpTransport	2968
6.674.3 Member Function Documentation	2968
6.674.3.1 close	2968
6.674.3.2 isClosed	2968
6.674.3.3 isConnected	2969
6.674.3.4 isFaultTolerant	2969
6.675activemq::transport::tcp::TcpTransportFactory Class Reference	2969
6.675.1 Detailed Description	2970
6.675.2 Constructor & Destructor Documentation	2970
6.675.2.1 ~TcpTransportFactory	2970
6.675.3 Member Function Documentation	2970
6.675.3.1 create	2970
6.675.3.2 createComposite	2970
6.675.3.3 doCreateComposite	2971
6.676cms::TemporaryQueue Class Reference	2971
6.676.1 Detailed Description	2972
6.676.2 Constructor & Destructor Documentation	2972
6.676.2.1 ~TemporaryQueue	2972
6.676.3 Member Function Documentation	2972
6.676.3.1 destroy	2972
6.676.3.2 getQueueName	2972
6.677cms::TemporaryTopic Class Reference	2973
6.677.1 Detailed Description	2973
6.677.2 Constructor & Destructor Documentation	2973
6.677.2.1 ~TemporaryTopic	2973
6.677.3 Member Function Documentation	2973
6.677.3.1 destroy	2973
6.677.3.2 getTopicName	2974
6.678cms::TextMessage Class Reference	2974
6.678.1 Detailed Description	2974
6.678.2 Constructor & Destructor Documentation	2975
6.678.2.1 ~TextMessage	2975
6.678.3 Member Function Documentation	2975
6.678.3.1 getText	2975
6.678.3.2 setText	2975
6.678.3.3 setText	2975

6.679	decaf::util::concurrent::ThreadFactory Class Reference	2976
6.679.1	Detailed Description	2976
6.679.2	Constructor & Destructor Documentation	2976
6.679.2.1	~ThreadFactory	2976
6.679.3	Member Function Documentation	2976
6.679.3.1	newThread	2976
6.680	decaf::lang::ThreadGroup Class Reference	2977
6.680.1	Detailed Description	2977
6.680.2	Constructor & Destructor Documentation	2977
6.680.2.1	ThreadGroup	2977
6.680.2.2	~ThreadGroup	2977
6.681	decaf::util::concurrent::ThreadPool Class Reference	2977
6.681.1	Detailed Description	2979
6.681.2	Member Typedef Documentation	2979
6.681.2.1	Task	2979
6.681.3	Constructor & Destructor Documentation	2979
6.681.3.1	ThreadPool	2979
6.681.3.2	~ThreadPool	2979
6.681.4	Member Function Documentation	2979
6.681.4.1	deQueueTask	2979
6.681.4.2	getBacklog	2980
6.681.4.3	getBlockSize	2980
6.681.4.4	getFreeThreadCount	2980
6.681.4.5	getInstance	2980
6.681.4.6	getMaxThreads	2980
6.681.4.7	getPoolSize	2981
6.681.4.8	onTaskCompleted	2981
6.681.4.9	onTaskException	2981
6.681.4.10	onTaskStarted	2981
6.681.4.11	queueTask	2982
6.681.4.12	reserve	2982
6.681.4.13	setBlockSize	2982
6.681.4.14	setMaxThreads	2982
6.681.5	Field Documentation	2983
6.681.5.1	DEFAULT_MAX_BLOCK_SIZE	2983
6.681.5.2	DEFAULT_MAX_POOL_SIZE	2983

6.682decaf::lang::Throwable Class Reference	2983
6.682.1 Detailed Description	2984
6.682.2 Constructor & Destructor Documentation	2984
6.682.2.1 Throwable	2984
6.682.2.2 ~Throwable	2984
6.682.3 Member Function Documentation	2984
6.682.3.1 clone	2984
6.682.3.2 getCause	2985
6.682.3.3 getMessage	2985
6.682.3.4 getStackTrace	2985
6.682.3.5 getStackTraceString	2986
6.682.3.6 initCause	2986
6.682.3.7 printStackTrace	2986
6.682.3.8 printStackTrace	2986
6.682.3.9 setMark	2986
6.683decaf::util::concurrent::TimeoutException Class Reference	2987
6.683.1 Constructor & Destructor Documentation	2987
6.683.1.1 TimeoutException	2987
6.683.1.2 TimeoutException	2988
6.683.1.3 TimeoutException	2988
6.683.1.4 TimeoutException	2988
6.683.1.5 TimeoutException	2988
6.683.1.6 TimeoutException	2989
6.683.1.7 ~TimeoutException	2989
6.683.2 Member Function Documentation	2989
6.683.2.1 clone	2989
6.684decaf::util::Timer Class Reference	2989
6.684.1 Detailed Description	2991
6.684.2 Constructor & Destructor Documentation	2992
6.684.2.1 Timer	2992
6.684.2.2 ~Timer	2992
6.684.3 Member Function Documentation	2992
6.684.3.1 cancel	2992
6.684.3.2 purge	2992
6.684.3.3 schedule	2992
6.684.3.4 schedule	2993

6.684.3.5	schedule	2993
6.684.3.6	schedule	2994
6.684.3.7	schedule	2995
6.684.3.8	schedule	2995
6.684.3.9	schedule	2996
6.684.3.10	schedule	2997
6.684.3.11	scheduleAtFixedRate	2997
6.684.3.12	scheduleAtFixedRate	2998
6.684.3.13	scheduleAtFixedRate	2999
6.684.3.14	scheduleAtFixedRate	2999
6.685	decaf::util::TimerTask Class Reference	3000
6.685.1	Detailed Description	3001
6.685.2	Constructor & Destructor Documentation	3001
6.685.2.1	TimerTask	3001
6.685.2.2	~TimerTask	3001
6.685.3	Member Function Documentation	3001
6.685.3.1	cancel	3001
6.685.3.2	getWhen	3002
6.685.3.3	isScheduled	3002
6.685.3.4	scheduledExecutionTime	3002
6.685.3.5	setScheduledTime	3002
6.685.4	Friends And Related Function Documentation	3002
6.685.4.1	decaf::internal::util::TimerTaskHeap	3002
6.685.4.2	Timer	3002
6.685.4.3	TimerImpl	3002
6.686	decaf::internal::util::TimerTaskHeap Class Reference	3002
6.686.1	Detailed Description	3003
6.686.2	Constructor & Destructor Documentation	3003
6.686.2.1	TimerTaskHeap	3003
6.686.2.2	~TimerTaskHeap	3003
6.686.3	Member Function Documentation	3003
6.686.3.1	adjustMinimum	3003
6.686.3.2	deleteIfCancelled	3004
6.686.3.3	find	3004
6.686.3.4	insert	3004
6.686.3.5	isEmpty	3004

6.686.3.6 peek	3004
6.686.3.7 remove	3004
6.686.3.8 reset	3005
6.686.3.9 size	3005
6.687decaf::util::concurrent::TimeUnit Class Reference	3005
6.687.1 Detailed Description	3007
6.687.2 Constructor & Destructor Documentation	3007
6.687.2.1 TimeUnit	3007
6.687.2.2 ~TimeUnit	3007
6.687.3 Member Function Documentation	3007
6.687.3.1 compareTo	3007
6.687.3.2 convert	3008
6.687.3.3 equals	3008
6.687.3.4 operator<	3009
6.687.3.5 operator==	3009
6.687.3.6 sleep	3009
6.687.3.7 timedJoin	3009
6.687.3.8 timedWait	3010
6.687.3.9 toDays	3010
6.687.3.10toHours	3011
6.687.3.11toMicros	3011
6.687.3.12toMillis	3011
6.687.3.13toMinutes	3012
6.687.3.14toNanos	3012
6.687.3.15toSeconds	3012
6.687.3.16toString	3013
6.687.3.17valueOf	3013
6.687.4 Field Documentation	3013
6.687.4.1 DAYS	3013
6.687.4.2 HOURS	3013
6.687.4.3 MICROSECONDS	3013
6.687.4.4 MILLISECONDS	3013
6.687.4.5 MINUTES	3013
6.687.4.6 NANOSECONDS	3013
6.687.4.7 SECONDS	3014
6.687.4.8 values	3014

6.688	cms::Topic Class Reference	3014
6.688.1	Detailed Description	3014
6.688.2	Constructor & Destructor Documentation	3014
6.688.2.1	~Topic	3014
6.688.3	Member Function Documentation	3014
6.688.3.1	getTopicName	3014
6.689	activemq::state::Tracked Class Reference	3015
6.689.1	Constructor & Destructor Documentation	3015
6.689.1.1	Tracked	3015
6.689.1.2	Tracked	3015
6.689.1.3	~Tracked	3015
6.689.2	Member Function Documentation	3015
6.689.2.1	isWaitingForResponse	3015
6.689.2.2	onResponse	3015
6.690	activemq::commands::TransactionId Class Reference	3015
6.690.1	Member Typedef Documentation	3017
6.690.1.1	COMPARATOR	3017
6.690.2	Constructor & Destructor Documentation	3017
6.690.2.1	TransactionId	3017
6.690.2.2	TransactionId	3017
6.690.2.3	~TransactionId	3017
6.690.3	Member Function Documentation	3017
6.690.3.1	cloneDataStructure	3017
6.690.3.2	compareTo	3017
6.690.3.3	copyDataStructure	3017
6.690.3.4	equals	3017
6.690.3.5	equals	3018
6.690.3.6	getDataStructureType	3018
6.690.3.7	operator<	3018
6.690.3.8	operator=	3018
6.690.3.9	operator==	3018
6.690.3.10	toString	3018
6.690.4	Field Documentation	3018
6.690.4.1	ID_TRANSACTIONID	3018
6.691	activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller Class Reference	3018
6.691.1	Detailed Description	3019

6.691.2 Constructor & Destructor Documentation	3020
6.691.2.1 TransactionIdMarshaller	3020
6.691.2.2 ~TransactionIdMarshaller	3020
6.691.3 Member Function Documentation	3020
6.691.3.1 looseMarshal	3020
6.691.3.2 looseUnmarshal	3020
6.691.3.3 tightMarshal1	3021
6.691.3.4 tightMarshal2	3021
6.691.3.5 tightUnmarshal	3022
6.692activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller Class Refer- ence	3022
6.692.1 Detailed Description	3023
6.692.2 Constructor & Destructor Documentation	3023
6.692.2.1 TransactionIdMarshaller	3023
6.692.2.2 ~TransactionIdMarshaller	3023
6.692.3 Member Function Documentation	3023
6.692.3.1 looseMarshal	3023
6.692.3.2 looseUnmarshal	3024
6.692.3.3 tightMarshal1	3024
6.692.3.4 tightMarshal2	3025
6.692.3.5 tightUnmarshal	3025
6.693activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller Class Refer- ence	3026
6.693.1 Detailed Description	3027
6.693.2 Constructor & Destructor Documentation	3027
6.693.2.1 TransactionIdMarshaller	3027
6.693.2.2 ~TransactionIdMarshaller	3027
6.693.3 Member Function Documentation	3027
6.693.3.1 looseMarshal	3027
6.693.3.2 looseUnmarshal	3027
6.693.3.3 tightMarshal1	3028
6.693.3.4 tightMarshal2	3028
6.693.3.5 tightUnmarshal	3029
6.694activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller Class Refer- ence	3029
6.694.1 Detailed Description	3030
6.694.2 Constructor & Destructor Documentation	3031

6.694.2.1 TransactionIdMarshaller	3031
6.694.2.2 ~TransactionIdMarshaller	3031
6.694.3 Member Function Documentation	3031
6.694.3.1 looseMarshal	3031
6.694.3.2 looseUnmarshal	3031
6.694.3.3 tightMarshal1	3032
6.694.3.4 tightMarshal2	3032
6.694.3.5 tightUnmarshal	3033
6.695activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller Class Reference	3033
6.695.1 Detailed Description	3034
6.695.2 Constructor & Destructor Documentation	3034
6.695.2.1 TransactionIdMarshaller	3034
6.695.2.2 ~TransactionIdMarshaller	3034
6.695.3 Member Function Documentation	3034
6.695.3.1 looseMarshal	3034
6.695.3.2 looseUnmarshal	3035
6.695.3.3 tightMarshal1	3035
6.695.3.4 tightMarshal2	3036
6.695.3.5 tightUnmarshal	3036
6.696activemq::commands::TransactionInfo Class Reference	3037
6.696.1 Constructor & Destructor Documentation	3038
6.696.1.1 TransactionInfo	3038
6.696.1.2 TransactionInfo	3038
6.696.1.3 ~TransactionInfo	3038
6.696.2 Member Function Documentation	3038
6.696.2.1 cloneDataStructure	3038
6.696.2.2 copyDataStructure	3038
6.696.2.3 equals	3039
6.696.2.4 getConnectionId	3039
6.696.2.5 getConnectionId	3039
6.696.2.6 getDataStructureType	3039
6.696.2.7 getTransactionId	3039
6.696.2.8 getTransactionId	3039
6.696.2.9 getType	3039
6.696.2.10sTransactionInfo	3039
6.696.2.11operator=	3040

6.696.2.12	setConnectionId	3040
6.696.2.13	setTransactionId	3040
6.696.2.14	setType	3040
6.696.2.15	toString	3040
6.696.2.16	visit	3040
6.696.3	Field Documentation	3040
6.696.3.1	connectionId	3040
6.696.3.2	ID_TRANSACTIONINFO	3040
6.696.3.3	transactionId	3040
6.696.3.4	type	3040
6.697	activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller Class Reference	3041
6.697.1	Detailed Description	3042
6.697.2	Constructor & Destructor Documentation	3042
6.697.2.1	TransactionInfoMarshaller	3042
6.697.2.2	~TransactionInfoMarshaller	3042
6.697.3	Member Function Documentation	3042
6.697.3.1	createObject	3042
6.697.3.2	getDataStructureType	3042
6.697.3.3	looseMarshal	3042
6.697.3.4	looseUnmarshal	3043
6.697.3.5	tightMarshal1	3043
6.697.3.6	tightMarshal2	3044
6.697.3.7	tightUnmarshal	3044
6.698	activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller Class Reference	3044
6.698.1	Detailed Description	3045
6.698.2	Constructor & Destructor Documentation	3046
6.698.2.1	TransactionInfoMarshaller	3046
6.698.2.2	~TransactionInfoMarshaller	3046
6.698.3	Member Function Documentation	3046
6.698.3.1	createObject	3046
6.698.3.2	getDataStructureType	3046
6.698.3.3	looseMarshal	3046
6.698.3.4	looseUnmarshal	3047
6.698.3.5	tightMarshal1	3047
6.698.3.6	tightMarshal2	3047

6.698.3.7 tightUnmarshal	3048
6.699activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller Class Reference	3048
6.699.1 Detailed Description	3049
6.699.2 Constructor & Destructor Documentation	3050
6.699.2.1 TransactionInfoMarshaller	3050
6.699.2.2 ~TransactionInfoMarshaller	3050
6.699.3 Member Function Documentation	3050
6.699.3.1 createObject	3050
6.699.3.2 getDataStructureType	3050
6.699.3.3 looseMarshal	3050
6.699.3.4 looseUnmarshal	3051
6.699.3.5 tightMarshal1	3051
6.699.3.6 tightMarshal2	3051
6.699.3.7 tightUnmarshal	3052
6.700activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller Class Reference	3052
6.700.1 Detailed Description	3053
6.700.2 Constructor & Destructor Documentation	3054
6.700.2.1 TransactionInfoMarshaller	3054
6.700.2.2 ~TransactionInfoMarshaller	3054
6.700.3 Member Function Documentation	3054
6.700.3.1 createObject	3054
6.700.3.2 getDataStructureType	3054
6.700.3.3 looseMarshal	3054
6.700.3.4 looseUnmarshal	3055
6.700.3.5 tightMarshal1	3055
6.700.3.6 tightMarshal2	3055
6.700.3.7 tightUnmarshal	3056
6.701activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller Class Reference	3056
6.701.1 Detailed Description	3057
6.701.2 Constructor & Destructor Documentation	3058
6.701.2.1 TransactionInfoMarshaller	3058
6.701.2.2 ~TransactionInfoMarshaller	3058
6.701.3 Member Function Documentation	3058
6.701.3.1 createObject	3058

6.701.3.2	getDataStructureType	3058
6.701.3.3	looseMarshal	3058
6.701.3.4	looseUnmarshal	3059
6.701.3.5	tightMarshal1	3059
6.701.3.6	tightMarshal2	3059
6.701.3.7	tightUnmarshal	3060
6.702	activemq::state::TransactionState Class Reference	3060
6.702.1	Constructor & Destructor Documentation	3061
6.702.1.1	TransactionState	3061
6.702.1.2	~TransactionState	3061
6.702.2	Member Function Documentation	3061
6.702.2.1	addCommand	3061
6.702.2.2	checkShutdown	3061
6.702.2.3	getCommands	3061
6.702.2.4	getId	3061
6.702.2.5	getPreparedResult	3061
6.702.2.6	isPrepared	3061
6.702.2.7	setPrepared	3061
6.702.2.8	setPreparedResult	3061
6.702.2.9	shutdown	3061
6.702.2.10	toString	3061
6.703	decaf::internal::util::concurrent::Transferer< E > Class Template Reference	3062
6.703.1	Detailed Description	3062
6.704	decaf::internal::util::concurrent::TransferQueue< E > Class Template Reference	3062
6.704.1	Detailed Description	3063
6.704.2	Constructor & Destructor Documentation	3063
6.704.2.1	TransferQueue	3063
6.704.2.2	~TransferQueue	3063
6.704.3	Member Function Documentation	3063
6.704.3.1	transfer	3063
6.704.3.2	transfer	3064
6.705	decaf::internal::util::concurrent::TransferStack< E > Class Template Reference	3064
6.705.1	Constructor & Destructor Documentation	3065
6.705.1.1	TransferStack	3065
6.705.1.2	~TransferStack	3065
6.705.2	Member Function Documentation	3065

6.705.2.1 transfer	3065
6.705.2.2 transfer	3065
6.706activemq::transport::Transport Class Reference	3066
6.706.1 Detailed Description	3067
6.706.2 Constructor & Destructor Documentation	3067
6.706.2.1 ~Transport	3067
6.706.3 Member Function Documentation	3067
6.706.3.1 getRemoteAddress	3067
6.706.3.2 getTransportListener	3068
6.706.3.3 isClosed	3068
6.706.3.4 isConnected	3068
6.706.3.5 isFaultTolerant	3068
6.706.3.6 narrow	3068
6.706.3.7 oneway	3069
6.706.3.8 reconnect	3069
6.706.3.9 request	3069
6.706.3.10request	3070
6.706.3.11setTransportListener	3070
6.706.3.12setWireFormat	3070
6.706.3.13start	3071
6.706.3.14stop	3071
6.707activemq::transport::TransportFactory Class Reference	3071
6.707.1 Detailed Description	3072
6.707.2 Constructor & Destructor Documentation	3072
6.707.2.1 ~TransportFactory	3072
6.707.3 Member Function Documentation	3072
6.707.3.1 create	3072
6.707.3.2 createComposite	3072
6.708activemq::transport::TransportFilter Class Reference	3073
6.708.1 Detailed Description	3075
6.708.2 Constructor & Destructor Documentation	3075
6.708.2.1 TransportFilter	3075
6.708.2.2 ~TransportFilter	3075
6.708.3 Member Function Documentation	3075
6.708.3.1 close	3075
6.708.3.2 fire	3075

6.708.3.3	fire	3076
6.708.3.4	getRemoteAddress	3076
6.708.3.5	getTransportListener	3076
6.708.3.6	isClosed	3076
6.708.3.7	isConnected	3076
6.708.3.8	isFaultTolerant	3077
6.708.3.9	narrow	3077
6.708.3.10	onCommand	3077
6.708.3.11	oneway	3077
6.708.3.12	onException	3078
6.708.3.13	reconnect	3078
6.708.3.14	request	3078
6.708.3.15	request	3079
6.708.3.16	setTransportListener	3079
6.708.3.17	setWireFormat	3079
6.708.3.18	start	3079
6.708.3.19	stop	3080
6.708.3.20	transportInterrupted	3080
6.708.3.21	transportResumed	3080
6.708.4	Field Documentation	3080
6.708.4.1	listener	3080
6.708.4.2	next	3080
6.709	activemq::transport::TransportListener Class Reference	3080
6.709.1	Detailed Description	3081
6.709.2	Constructor & Destructor Documentation	3081
6.709.2.1	~TransportListener	3081
6.709.3	Member Function Documentation	3081
6.709.3.1	onCommand	3081
6.709.3.2	onException	3082
6.709.3.3	transportInterrupted	3082
6.709.3.4	transportResumed	3082
6.710	activemq::transport::TransportRegistry Class Reference	3082
6.710.1	Detailed Description	3083
6.710.2	Constructor & Destructor Documentation	3083
6.710.2.1	~TransportRegistry	3083
6.710.3	Member Function Documentation	3083

6.710.3.1 findFactory	3083
6.710.3.2 getInstance	3084
6.710.3.3 getTransportNames	3084
6.710.3.4 registerFactory	3084
6.710.3.5 unregisterFactory	3084
6.711decaf::net::UnknownHostException Class Reference	3085
6.711.1 Constructor & Destructor Documentation	3085
6.711.1.1 UnknownHostException	3085
6.711.1.2 UnknownHostException	3086
6.711.1.3 UnknownHostException	3086
6.711.1.4 UnknownHostException	3086
6.711.1.5 UnknownHostException	3086
6.711.1.6 UnknownHostException	3087
6.711.1.7 ~UnknownHostException	3087
6.711.2 Member Function Documentation	3087
6.711.2.1 clone	3087
6.712decaf::net::UnknownServiceException Class Reference	3087
6.712.1 Constructor & Destructor Documentation	3088
6.712.1.1 UnknownServiceException	3088
6.712.1.2 UnknownServiceException	3088
6.712.1.3 UnknownServiceException	3088
6.712.1.4 UnknownServiceException	3089
6.712.1.5 UnknownServiceException	3089
6.712.1.6 UnknownServiceException	3089
6.712.1.7 ~UnknownServiceException	3089
6.712.2 Member Function Documentation	3089
6.712.2.1 clone	3089
6.713decaf::io::UnsupportedEncodingException Class Reference	3090
6.713.1 Detailed Description	3091
6.713.2 Constructor & Destructor Documentation	3091
6.713.2.1 UnsupportedEncodingException	3091
6.713.2.2 UnsupportedEncodingException	3091
6.713.2.3 UnsupportedEncodingException	3091
6.713.2.4 UnsupportedEncodingException	3091
6.713.2.5 UnsupportedEncodingException	3092
6.713.2.6 UnsupportedEncodingException	3092

6.713.2.7 ~UnsupportedEncodingException	3092
6.713.3 Member Function Documentation	3092
6.713.3.1 clone	3092
6.714cms::UnsupportedOperationException Class Reference	3093
6.714.1 Detailed Description	3093
6.714.2 Constructor & Destructor Documentation	3093
6.714.2.1 UnsupportedOperationException	3093
6.714.2.2 UnsupportedOperationException	3093
6.714.2.3 UnsupportedOperationException	3093
6.714.2.4 UnsupportedOperationException	3093
6.714.2.5 ~UnsupportedOperationException	3093
6.715decaf::lang::exceptions::UnsupportedOperationException Class Reference	3094
6.715.1 Constructor & Destructor Documentation	3094
6.715.1.1 UnsupportedOperationException	3094
6.715.1.2 UnsupportedOperationException	3095
6.715.1.3 UnsupportedOperationException	3095
6.715.1.4 UnsupportedOperationException	3095
6.715.1.5 UnsupportedOperationException	3095
6.715.1.6 UnsupportedOperationException	3095
6.715.1.7 ~UnsupportedOperationException	3096
6.715.2 Member Function Documentation	3096
6.715.2.1 clone	3096
6.716decaf::net::URI Class Reference	3096
6.716.1 Detailed Description	3099
6.716.2 Constructor & Destructor Documentation	3099
6.716.2.1 URI	3099
6.716.2.2 URI	3099
6.716.2.3 URI	3099
6.716.2.4 URI	3099
6.716.2.5 URI	3099
6.716.2.6 URI	3100
6.716.2.7 URI	3100
6.716.2.8 ~URI	3100
6.716.3 Member Function Documentation	3100
6.716.3.1 compareTo	3100
6.716.3.2 create	3101

6.716.3.3 equals	3101
6.716.3.4 getAuthority	3101
6.716.3.5 getFragment	3101
6.716.3.6 getHost	3101
6.716.3.7 getPath	3102
6.716.3.8 getPort	3102
6.716.3.9 getQuery	3102
6.716.3.10 getRawAuthority	3102
6.716.3.11 getRawFragment	3102
6.716.3.12 getRawPath	3102
6.716.3.13 getRawQuery	3103
6.716.3.14 getRawSchemeSpecificPart	3103
6.716.3.15 getRawUserInfo	3103
6.716.3.16 getScheme	3103
6.716.3.17 getSchemeSpecificPart	3103
6.716.3.18 getUserInfo	3103
6.716.3.19 isAbsolute	3104
6.716.3.20 isOpaque	3104
6.716.3.21 normalize	3104
6.716.3.22 operator<	3104
6.716.3.23 operator==	3105
6.716.3.24 parseServerAuthority	3105
6.716.3.25 relativize	3105
6.716.3.26 resolve	3106
6.716.3.27 resolve	3106
6.716.3.28 toString	3107
6.716.3.29 toURL	3107
6.717.decaf::internal::net::URLEncoderDecoder Class Reference	3107
6.717.1 Constructor & Destructor Documentation	3108
6.717.1.1 URLEncoderDecoder	3108
6.717.1.2 ~URLEncoderDecoder	3108
6.717.2 Member Function Documentation	3108
6.717.2.1 decode	3108
6.717.2.2 encodeOthers	3109
6.717.2.3 quoteIllegal	3109
6.717.2.4 validate	3109

6.717.2.5 validateSimple	3110
6.718decaf::internal::net::URIHelper Class Reference	3110
6.718.1 Detailed Description	3111
6.718.2 Constructor & Destructor Documentation	3112
6.718.2.1 URIHelper	3112
6.718.2.2 URIHelper	3112
6.718.2.3 ~URIHelper	3112
6.718.3 Member Function Documentation	3112
6.718.3.1 isValidDomainName	3112
6.718.3.2 isValidHexChar	3112
6.718.3.3 isValidHost	3113
6.718.3.4 isValidIP4Word	3113
6.718.3.5 isValidIP6Address	3113
6.718.3.6 isValidIPv4Address	3113
6.718.3.7 parseAuthority	3114
6.718.3.8 parseURI	3114
6.718.3.9 validateAuthority	3115
6.718.3.10validateFragment	3115
6.718.3.11validatePath	3115
6.718.3.12validateQuery	3116
6.718.3.13validateScheme	3116
6.718.3.14validateSsp	3116
6.718.3.15validateUserinfo	3117
6.719activemq::transport::failover::URIPool Class Reference	3117
6.719.1 Constructor & Destructor Documentation	3118
6.719.1.1 URIPool	3118
6.719.1.2 URIPool	3118
6.719.1.3 ~URIPool	3118
6.719.2 Member Function Documentation	3118
6.719.2.1 addURI	3118
6.719.2.2 addURIs	3118
6.719.2.3 getURI	3119
6.719.2.4 isRandomize	3119
6.719.2.5 removeURI	3119
6.719.2.6 setRandomize	3119
6.720activemq::util::URISupport Class Reference	3119

6.720.1 Member Function Documentation	3120
6.720.1.1 createQueryString	3120
6.720.1.2 parseComposite	3121
6.720.1.3 parseQuery	3121
6.720.1.4 parseQuery	3121
6.720.1.5 parseURL	3122
6.721decaf::net::URISyntaxException Class Reference	3122
6.721.1 Constructor & Destructor Documentation	3123
6.721.1.1 URISyntaxException	3123
6.721.1.2 URISyntaxException	3123
6.721.1.3 URISyntaxException	3123
6.721.1.4 URISyntaxException	3123
6.721.1.5 URISyntaxException	3124
6.721.1.6 URISyntaxException	3124
6.721.1.7 URISyntaxException	3124
6.721.1.8 URISyntaxException	3125
6.721.1.9 ~URISyntaxException	3125
6.721.2 Member Function Documentation	3125
6.721.2.1 clone	3125
6.721.2.2 getIndex	3125
6.721.2.3 getInput	3125
6.721.2.4 getReason	3126
6.722decaf::internal::net::URIType Class Reference	3126
6.722.1 Detailed Description	3128
6.722.2 Constructor & Destructor Documentation	3128
6.722.2.1 URIType	3128
6.722.2.2 URIType	3128
6.722.2.3 ~URIType	3128
6.722.3 Member Function Documentation	3128
6.722.3.1 getAuthority	3128
6.722.3.2 getFragment	3128
6.722.3.3 getHost	3128
6.722.3.4 getPath	3129
6.722.3.5 getPort	3129
6.722.3.6 getQuery	3129
6.722.3.7 getScheme	3129

6.722.3.8	getSchemeSpecificPart	3129
6.722.3.9	getSource	3129
6.722.3.10	getUserInfo	3130
6.722.3.11	isAbsolute	3130
6.722.3.12	isOpaque	3130
6.722.3.13	isServerAuthority	3130
6.722.3.14	isValid	3130
6.722.3.15	setAbsolute	3130
6.722.3.16	setAuthority	3131
6.722.3.17	setFragment	3131
6.722.3.18	setHost	3131
6.722.3.19	setOpaque	3131
6.722.3.20	setPath	3131
6.722.3.21	setPort	3131
6.722.3.22	setQuery	3132
6.722.3.23	setScheme	3132
6.722.3.24	setSchemeSpecificPart	3132
6.722.3.25	setServerAuthority	3132
6.722.3.26	setSource	3132
6.722.3.27	setUserInfo	3133
6.722.3.28	setValid	3133
6.723	decaf::net::URL Class Reference	3133
6.723.1	Detailed Description	3133
6.723.2	Constructor & Destructor Documentation	3135
6.723.2.1	URL	3135
6.723.2.2	URL	3135
6.723.2.3	~URL	3135
6.724	decaf::net::URLDecoder Class Reference	3135
6.724.1	Constructor & Destructor Documentation	3135
6.724.1.1	~URLDecoder	3135
6.724.2	Member Function Documentation	3135
6.724.2.1	decode	3135
6.725	decaf::net::URLEncoder Class Reference	3136
6.725.1	Constructor & Destructor Documentation	3136
6.725.1.1	~URLEncoder	3136
6.725.2	Member Function Documentation	3136

6.725.2.1 encode	3136
6.726activemq::util::Usage Class Reference	3137
6.726.1 Constructor & Destructor Documentation	3137
6.726.1.1 ~Usage	3137
6.726.2 Member Function Documentation	3137
6.726.2.1 decreaseUsage	3137
6.726.2.2 enqueueUsage	3138
6.726.2.3 increaseUsage	3138
6.726.2.4 isFull	3138
6.726.2.5 waitForSpace	3138
6.726.2.6 waitForSpace	3138
6.727decaf::io::UTFDataFormatException Class Reference	3139
6.727.1 Detailed Description	3139
6.727.2 Constructor & Destructor Documentation	3139
6.727.2.1 UTFDataFormatException	3139
6.727.2.2 UTFDataFormatException	3140
6.727.2.3 UTFDataFormatException	3140
6.727.2.4 UTFDataFormatException	3140
6.727.2.5 UTFDataFormatException	3140
6.727.2.6 UTFDataFormatException	3140
6.727.2.7 ~UTFDataFormatException	3141
6.727.3 Member Function Documentation	3141
6.727.3.1 clone	3141
6.728decaf::util::UUID Class Reference	3141
6.728.1 Detailed Description	3143
6.728.2 Constructor & Destructor Documentation	3143
6.728.2.1 UUID	3143
6.728.2.2 ~UUID	3143
6.728.3 Member Function Documentation	3143
6.728.3.1 clockSequence	3143
6.728.3.2 compareTo	3144
6.728.3.3 equals	3144
6.728.3.4 fromString	3144
6.728.3.5 getLeastSignificantBits	3144
6.728.3.6 getMostSignificantBits	3145
6.728.3.7 nameUUIDFromBytes	3145

6.728.3.8 nameUUIDFromBytes	3145
6.728.3.9 node	3145
6.728.3.10 operator<	3146
6.728.3.11 operator==	3146
6.728.3.12 randomUUID	3146
6.728.3.13 timestamp	3146
6.728.3.14 toString	3147
6.728.3.15 variant	3147
6.728.3.16 version	3147
6.729 activemq::wireformat::WireFormat Class Reference	3148
6.729.1 Detailed Description	3148
6.729.2 Constructor & Destructor Documentation	3149
6.729.2.1 ~WireFormat	3149
6.729.3 Member Function Documentation	3149
6.729.3.1 createNegotiator	3149
6.729.3.2 getVersion	3149
6.729.3.3 hasNegotiator	3149
6.729.3.4 inReceive	3150
6.729.3.5 marshal	3150
6.729.3.6 setVersion	3150
6.729.3.7 unmarshal	3151
6.730 activemq::wireformat::WireFormatFactory Class Reference	3151
6.730.1 Detailed Description	3152
6.730.2 Constructor & Destructor Documentation	3152
6.730.2.1 ~WireFormatFactory	3152
6.730.3 Member Function Documentation	3152
6.730.3.1 createWireFormat	3152
6.731 activemq::commands::WireFormatInfo Class Reference	3152
6.731.1 Constructor & Destructor Documentation	3155
6.731.1.1 WireFormatInfo	3155
6.731.1.2 ~WireFormatInfo	3155
6.731.2 Member Function Documentation	3155
6.731.2.1 afterUnmarshal	3155
6.731.2.2 beforeMarshal	3155
6.731.2.3 cloneDataStructure	3156
6.731.2.4 copyDataStructure	3156

6.731.2.5 equals	3156
6.731.2.6 getCacheSize	3156
6.731.2.7 getDataStructureType	3156
6.731.2.8 getMagic	3157
6.731.2.9 getMarshaledProperties	3157
6.731.2.10 getMaxInactivityDuration	3157
6.731.2.11 getMaxInactivityDurationInitialDelay	3157
6.731.2.12 getProperties	3158
6.731.2.13 getProperties	3158
6.731.2.14 getVersion	3158
6.731.2.15 isCacheEnabled	3158
6.731.2.16 isMarshalAware	3158
6.731.2.17 isSizePrefixDisabled	3159
6.731.2.18 isStackTraceEnabled	3159
6.731.2.19 isTcpNoDelayEnabled	3159
6.731.2.20 isTightEncodingEnabled	3159
6.731.2.21 isValid	3159
6.731.2.22 isWireFormatInfo	3159
6.731.2.23 setCacheEnabled	3160
6.731.2.24 setCacheSize	3160
6.731.2.25 setMagic	3160
6.731.2.26 setMarshaledProperties	3160
6.731.2.27 setMaxInactivityDuration	3160
6.731.2.28 setMaxInactivityDurationInitialDelay	3161
6.731.2.29 setProperties	3161
6.731.2.30 setSizePrefixDisabled	3161
6.731.2.31 setStackTraceEnabled	3161
6.731.2.32 setTcpNoDelayEnabled	3161
6.731.2.33 setTightEncodingEnabled	3162
6.731.2.34 setVersion	3162
6.731.2.35 toString	3162
6.731.2.36 visit	3162
6.731.3 Field Documentation	3162
6.731.3.1 ID_WIREFORMATINFO	3162
6.732 activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller Class Reference	3163
6.732.1 Detailed Description	3164

6.732.2 Constructor & Destructor Documentation	3164
6.732.2.1 WireFormatInfoMarshaller	3164
6.732.2.2 ~WireFormatInfoMarshaller	3164
6.732.3 Member Function Documentation	3164
6.732.3.1 createObject	3164
6.732.3.2 getDataStructureType	3164
6.732.3.3 looseMarshal	3164
6.732.3.4 looseUnmarshal	3165
6.732.3.5 tightMarshal1	3165
6.732.3.6 tightMarshal2	3166
6.732.3.7 tightUnmarshal	3166
6.733activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller Class Reference	3166
6.733.1 Detailed Description	3167
6.733.2 Constructor & Destructor Documentation	3168
6.733.2.1 WireFormatInfoMarshaller	3168
6.733.2.2 ~WireFormatInfoMarshaller	3168
6.733.3 Member Function Documentation	3168
6.733.3.1 createObject	3168
6.733.3.2 getDataStructureType	3168
6.733.3.3 looseMarshal	3168
6.733.3.4 looseUnmarshal	3169
6.733.3.5 tightMarshal1	3169
6.733.3.6 tightMarshal2	3169
6.733.3.7 tightUnmarshal	3170
6.734activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller Class Reference	3170
6.734.1 Detailed Description	3171
6.734.2 Constructor & Destructor Documentation	3172
6.734.2.1 WireFormatInfoMarshaller	3172
6.734.2.2 ~WireFormatInfoMarshaller	3172
6.734.3 Member Function Documentation	3172
6.734.3.1 createObject	3172
6.734.3.2 getDataStructureType	3172
6.734.3.3 looseMarshal	3172
6.734.3.4 looseUnmarshal	3173
6.734.3.5 tightMarshal1	3173

6.734.3.6 tightMarshal2	3173
6.734.3.7 tightUnmarshal	3174
6.735activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller Class Reference	3174
6.735.1 Detailed Description	3175
6.735.2 Constructor & Destructor Documentation	3176
6.735.2.1 WireFormatInfoMarshaller	3176
6.735.2.2 ~WireFormatInfoMarshaller	3176
6.735.3 Member Function Documentation	3176
6.735.3.1 createObject	3176
6.735.3.2 getDataStructureType	3176
6.735.3.3 looseMarshal	3176
6.735.3.4 looseUnmarshal	3177
6.735.3.5 tightMarshal1	3177
6.735.3.6 tightMarshal2	3177
6.735.3.7 tightUnmarshal	3178
6.736activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller Class Reference	3178
6.736.1 Detailed Description	3179
6.736.2 Constructor & Destructor Documentation	3180
6.736.2.1 WireFormatInfoMarshaller	3180
6.736.2.2 ~WireFormatInfoMarshaller	3180
6.736.3 Member Function Documentation	3180
6.736.3.1 createObject	3180
6.736.3.2 getDataStructureType	3180
6.736.3.3 looseMarshal	3180
6.736.3.4 looseUnmarshal	3181
6.736.3.5 tightMarshal1	3181
6.736.3.6 tightMarshal2	3181
6.736.3.7 tightUnmarshal	3182
6.737activemq::wireformat::WireFormatNegotiator Class Reference	3182
6.737.1 Detailed Description	3183
6.737.2 Constructor & Destructor Documentation	3183
6.737.2.1 WireFormatNegotiator	3183
6.737.2.2 ~WireFormatNegotiator	3183
6.738activemq::wireformat::WireFormatRegistry Class Reference	3183
6.738.1 Detailed Description	3184

6.738.2 Constructor & Destructor Documentation	3184
6.738.2.1 ~WireFormatRegistry	3184
6.738.3 Member Function Documentation	3184
6.738.3.1 findFactory	3184
6.738.3.2 getInstance	3185
6.738.3.3 getWireFormatNames	3185
6.738.3.4 registerFactory	3185
6.738.3.5 unregisterFactory	3185
6.739activemq::transport::inactivity::WriteChecker Class Reference	3186
6.739.1 Detailed Description	3186
6.739.2 Constructor & Destructor Documentation	3186
6.739.2.1 WriteChecker	3186
6.739.2.2 ~WriteChecker	3186
6.739.3 Member Function Documentation	3186
6.739.3.1 run	3186
6.740decaf::io::Writer Class Reference	3187
6.740.1 Constructor & Destructor Documentation	3187
6.740.1.1 ~Writer	3187
6.740.2 Member Function Documentation	3187
6.740.2.1 getOutputStream	3187
6.740.2.2 setOutputStream	3187
6.740.2.3 write	3188
6.740.2.4 writeByte	3188
6.741decaf::security::auth::x500::X500Principal Class Reference	3188
6.741.1 Constructor & Destructor Documentation	3189
6.741.1.1 ~X500Principal	3189
6.741.2 Member Function Documentation	3189
6.741.2.1 getEncoded	3189
6.741.2.2 getName	3189
6.741.2.3 hashCode	3189
6.742decaf::security::cert::X509Certificate Class Reference	3189
6.742.1 Detailed Description	3190
6.742.2 Constructor & Destructor Documentation	3190
6.742.2.1 ~X509Certificate	3190
6.742.3 Member Function Documentation	3190
6.742.3.1 checkValidity	3190

6.742.3.2	checkValidity	3190
6.742.3.3	getBasicConstraints	3190
6.742.3.4	getIssuerUniqueID	3190
6.742.3.5	getIssuerX500Principal	3191
6.742.3.6	getKeyUsage	3191
6.742.3.7	getNotAfter	3191
6.742.3.8	getNotBefore	3191
6.742.3.9	getSigAlgName	3191
6.742.3.10	getSigAlgOID	3191
6.742.3.11	getSigAlgParams	3191
6.742.3.12	getSignature	3191
6.742.3.13	getSubjectUniqueID	3192
6.742.3.14	getSubjectX500Principal	3192
6.742.3.15	getTBSCertificate	3192
6.742.3.16	getVersion	3192
6.743	activemq::commands::XATransactionId Class Reference	3192
6.743.1	Member Typedef Documentation	3194
6.743.1.1	COMPARATOR	3194
6.743.2	Constructor & Destructor Documentation	3194
6.743.2.1	XATransactionId	3194
6.743.2.2	XATransactionId	3194
6.743.2.3	~XATransactionId	3194
6.743.3	Member Function Documentation	3194
6.743.3.1	cloneDataStructure	3194
6.743.3.2	compareTo	3194
6.743.3.3	copyDataStructure	3194
6.743.3.4	equals	3194
6.743.3.5	equals	3195
6.743.3.6	getBranchQualifier	3195
6.743.3.7	getBranchQualifier	3195
6.743.3.8	getDataStructureType	3195
6.743.3.9	getFormatId	3196
6.743.3.10	getGlobalTransactionId	3196
6.743.3.11	getGlobalTransactionId	3196
6.743.3.12	operator<	3196
6.743.3.13	operator=	3196

6.743.3.14	operator==	3196
6.743.3.15	setBranchQualifier	3196
6.743.3.16	setFormatId	3196
6.743.3.17	setGlobalTransactionId	3196
6.743.3.18	toString	3196
6.743.4	Field Documentation	3197
6.743.4.1	branchQualifier	3197
6.743.4.2	formatId	3197
6.743.4.3	globalTransactionId	3197
6.743.4.4	ID_XATRANSSACTIONID	3197
6.744	activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller Class	
	Reference	3197
6.744.1	Detailed Description	3198
6.744.2	Constructor & Destructor Documentation	3198
6.744.2.1	XATransactionIdMarshaller	3198
6.744.2.2	~XATransactionIdMarshaller	3198
6.744.3	Member Function Documentation	3198
6.744.3.1	createObject	3198
6.744.3.2	getDataStructureType	3198
6.744.3.3	looseMarshal	3199
6.744.3.4	looseUnmarshal	3199
6.744.3.5	tightMarshal1	3199
6.744.3.6	tightMarshal2	3200
6.744.3.7	tightUnmarshal	3200
6.745	activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller Class	
	Reference	3201
6.745.1	Detailed Description	3202
6.745.2	Constructor & Destructor Documentation	3202
6.745.2.1	XATransactionIdMarshaller	3202
6.745.2.2	~XATransactionIdMarshaller	3202
6.745.3	Member Function Documentation	3202
6.745.3.1	createObject	3202
6.745.3.2	getDataStructureType	3202
6.745.3.3	looseMarshal	3203
6.745.3.4	looseUnmarshal	3203
6.745.3.5	tightMarshal1	3203
6.745.3.6	tightMarshal2	3204

6.745.3.7 tightUnmarshal	3204
6.746activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller Class	
Reference	3205
6.746.1 Detailed Description	3206
6.746.2 Constructor & Destructor Documentation	3206
6.746.2.1 XATransactionIdMarshaller	3206
6.746.2.2 ~XATransactionIdMarshaller	3206
6.746.3 Member Function Documentation	3206
6.746.3.1 createObject	3206
6.746.3.2 getDataStructureType	3206
6.746.3.3 looseMarshal	3207
6.746.3.4 looseUnmarshal	3207
6.746.3.5 tightMarshal1	3207
6.746.3.6 tightMarshal2	3208
6.746.3.7 tightUnmarshal	3208
6.747activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller Class	
Reference	3209
6.747.1 Detailed Description	3210
6.747.2 Constructor & Destructor Documentation	3210
6.747.2.1 XATransactionIdMarshaller	3210
6.747.2.2 ~XATransactionIdMarshaller	3210
6.747.3 Member Function Documentation	3210
6.747.3.1 createObject	3210
6.747.3.2 getDataStructureType	3210
6.747.3.3 looseMarshal	3211
6.747.3.4 looseUnmarshal	3211
6.747.3.5 tightMarshal1	3211
6.747.3.6 tightMarshal2	3212
6.747.3.7 tightUnmarshal	3212
6.748activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller Class	
Reference	3213
6.748.1 Detailed Description	3214
6.748.2 Constructor & Destructor Documentation	3214
6.748.2.1 XATransactionIdMarshaller	3214
6.748.2.2 ~XATransactionIdMarshaller	3214
6.748.3 Member Function Documentation	3214
6.748.3.1 createObject	3214

6.748.3.2	getDataStructureType	3214
6.748.3.3	looseMarshal	3215
6.748.3.4	looseUnmarshal	3215
6.748.3.5	tightMarshal1	3215
6.748.3.6	tightMarshal2	3216
6.748.3.7	tightUnmarshal	3216
7	File Documentation	3219
7.1	src/main/activemq/cmsutil/CachedConsumer.h File Reference	3219
7.2	src/main/activemq/cmsutil/CachedProducer.h File Reference	3219
7.3	src/main/activemq/cmsutil/CmsAccessor.h File Reference	3220
7.4	src/main/activemq/cmsutil/CmsDestinationAccessor.h File Reference	3220
7.5	src/main/activemq/cmsutil/CmsTemplate.h File Reference	3221
7.6	src/main/activemq/cmsutil/DestinationResolver.h File Reference	3221
7.7	src/main/activemq/cmsutil/DynamicDestinationResolver.h File Reference	3222
7.8	src/main/activemq/cmsutil/MessageCreator.h File Reference	3222
7.9	src/main/activemq/cmsutil/PooledSession.h File Reference	3223
7.10	src/main/activemq/cmsutil/ProducerCallback.h File Reference	3223
7.11	src/main/activemq/cmsutil/ResourceLifecycleManager.h File Reference	3224
7.12	src/main/activemq/cmsutil/SessionCallback.h File Reference	3225
7.13	src/main/activemq/cmsutil/SessionPool.h File Reference	3225
7.14	src/main/activemq/commands/ActiveMQBlobMessage.h File Reference	3226
7.15	src/main/activemq/commands/ActiveMQBytesMessage.h File Reference	3226
7.16	src/main/activemq/commands/ActiveMQDestination.h File Reference	3227
7.17	src/main/activemq/commands/ActiveMQMapMessage.h File Reference	3227
7.18	src/main/activemq/commands/ActiveMQMessage.h File Reference	3228
7.19	src/main/activemq/commands/ActiveMQMessageTemplate.h File Reference	3228
7.20	src/main/activemq/commands/ActiveMQObjectMessage.h File Reference	3229
7.21	src/main/activemq/commands/ActiveMQQueue.h File Reference	3229
7.22	src/main/activemq/commands/ActiveMQStreamMessage.h File Reference	3230
7.23	src/main/activemq/commands/ActiveMQTempDestination.h File Reference	3231
7.24	src/main/activemq/commands/ActiveMQTempQueue.h File Reference	3231
7.25	src/main/activemq/commands/ActiveMQTempTopic.h File Reference	3232
7.26	src/main/activemq/commands/ActiveMQTextMessage.h File Reference	3232
7.27	src/main/activemq/commands/ActiveMQTopic.h File Reference	3233
7.28	src/main/activemq/commands/BaseCommand.h File Reference	3233
7.29	src/main/activemq/commands/BaseDataStructure.h File Reference	3234

7.30	src/main/activemq/commands/BooleanExpression.h File Reference	3234
7.31	src/main/activemq/commands/BrokerError.h File Reference	3235
7.32	src/main/activemq/commands/BrokerId.h File Reference	3235
7.33	src/main/activemq/commands/BrokerInfo.h File Reference	3236
7.34	src/main/activemq/commands/Command.h File Reference	3236
7.35	src/main/activemq/commands/ConnectionControl.h File Reference	3237
7.36	src/main/activemq/commands/ConnectionError.h File Reference	3237
7.37	src/main/activemq/commands/ConnectionId.h File Reference	3238
7.38	src/main/activemq/commands/ConnectionInfo.h File Reference	3238
7.39	src/main/activemq/commands/ConsumerControl.h File Reference	3239
7.40	src/main/activemq/commands/ConsumerId.h File Reference	3239
7.41	src/main/activemq/commands/ConsumerInfo.h File Reference	3240
7.42	src/main/activemq/commands/ControlCommand.h File Reference	3241
7.43	src/main/activemq/commands/DataArrayResponse.h File Reference	3241
7.44	src/main/activemq/commands/DataResponse.h File Reference	3242
7.45	src/main/activemq/commands/DataStructure.h File Reference	3242
7.46	src/main/activemq/commands/DestinationInfo.h File Reference	3242
7.47	src/main/activemq/commands/DiscoveryEvent.h File Reference	3243
7.48	src/main/activemq/commands/ExceptionResponse.h File Reference	3244
7.49	src/main/activemq/commands/FlushCommand.h File Reference	3244
7.50	src/main/activemq/commands/IntegerResponse.h File Reference	3245
7.51	src/main/activemq/commands/JournalQueueAck.h File Reference	3245
7.52	src/main/activemq/commands/JournalTopicAck.h File Reference	3246
7.53	src/main/activemq/commands/JournalTrace.h File Reference	3246
7.54	src/main/activemq/commands/JournalTransaction.h File Reference	3247
7.55	src/main/activemq/commands/KeepAliveInfo.h File Reference	3247
7.56	src/main/activemq/commands/LastPartialCommand.h File Reference	3248
7.57	src/main/activemq/commands/LocalTransactionId.h File Reference	3248
7.58	src/main/activemq/commands/Message.h File Reference	3249
7.59	src/main/cms/Message.h File Reference	3249
7.60	src/main/activemq/commands/MessageAck.h File Reference	3250
7.61	src/main/activemq/commands/MessageDispatch.h File Reference	3251
7.62	src/main/activemq/commands/MessageDispatchNotification.h File Reference . . .	3251
7.63	src/main/activemq/commands/MessageId.h File Reference	3252
7.64	src/main/activemq/commands/MessagePull.h File Reference	3252
7.65	src/main/activemq/commands/NetworkBridgeFilter.h File Reference	3253

7.66	src/main/activemq/commands/PartialCommand.h File Reference	3253
7.67	src/main/activemq/commands/ProducerAck.h File Reference	3254
7.68	src/main/activemq/commands/ProducerId.h File Reference	3254
7.69	src/main/activemq/commands/ProducerInfo.h File Reference	3255
7.70	src/main/activemq/commands/RemoveInfo.h File Reference	3255
7.71	src/main/activemq/commands/RemoveSubscriptionInfo.h File Reference	3256
7.72	src/main/activemq/commands/ReplayCommand.h File Reference	3256
7.73	src/main/activemq/commands/Response.h File Reference	3257
7.74	src/main/activemq/commands/SessionId.h File Reference	3257
7.75	src/main/activemq/commands/SessionInfo.h File Reference	3258
7.76	src/main/activemq/commands/ShutdownInfo.h File Reference	3259
7.77	src/main/activemq/commands/SubscriptionInfo.h File Reference	3259
7.78	src/main/activemq/commands/TransactionId.h File Reference	3260
7.79	src/main/activemq/commands/TransactionInfo.h File Reference	3260
7.80	src/main/activemq/commands/WireFormatInfo.h File Reference	3261
7.81	src/main/activemq/commands/XATransactionId.h File Reference	3261
7.82	src/main/activemq/core/ActiveMQAckHandler.h File Reference	3262
7.83	src/main/activemq/core/ActiveMQConnection.h File Reference	3262
7.84	src/main/activemq/core/ActiveMQConnectionFactory.h File Reference	3263
7.85	src/main/activemq/core/ActiveMQConnectionMetaData.h File Reference	3264
7.86	src/main/activemq/core/ActiveMQConnectionSupport.h File Reference	3264
7.87	src/main/activemq/core/ActiveMQConstants.h File Reference	3265
7.88	src/main/activemq/core/ActiveMQConsumer.h File Reference	3265
7.89	src/main/activemq/core/ActiveMQProducer.h File Reference	3266
7.90	src/main/activemq/core/ActiveMQSession.h File Reference	3267
7.91	src/main/activemq/core/ActiveMQSessionExecutor.h File Reference	3268
7.92	src/main/activemq/core/ActiveMQTransactionContext.h File Reference	3268
7.93	src/main/activemq/core/DispatchData.h File Reference	3269
7.94	src/main/activemq/core/Dispatcher.h File Reference	3269
7.95	src/main/activemq/core/MessageDispatchChannel.h File Reference	3270
7.96	src/main/activemq/core/Synchronization.h File Reference	3270
7.97	src/main/activemq/exceptions/ActiveMQException.h File Reference	3271
7.98	src/main/activemq/exceptions/BrokerException.h File Reference	3271
7.99	src/main/activemq/exceptions/ExceptionDefines.h File Reference	3272
7.99.1	Define Documentation	3272
7.99.1.1	AMQ_CATCH_EXCEPTION_CONVERT	3272

7.99.1.2	AMQ_CATCH_NOTHROW	3273
7.99.1.3	AMQ_CATCH_RETHROW	3273
7.99.1.4	AMQ_CATCHALL_NOTHROW	3273
7.99.1.5	AMQ_CATCHALL_THROW	3274
7.100	src/main/decaf/lang/exceptions/ExceptionDefines.h File Reference	3274
7.100.1	Define Documentation	3274
7.100.1.1	DECAF_CATCH_EXCEPTION_CONVERT	3274
7.100.1.2	DECAF_CATCH_NOTHROW	3275
7.100.1.3	DECAF_CATCH_RETHROW	3275
7.100.1.4	DECAF_CATCHALL_NOTHROW	3275
7.100.1.5	DECAF_CATCHALL_THROW	3276
7.101	src/main/activemq/io/LoggingInputStream.h File Reference	3276
7.102	src/main/activemq/io/LoggingOutputStream.h File Reference	3277
7.103	src/main/activemq/library/ActiveMQCPP.h File Reference	3277
7.104	src/main/activemq/state/CommandVisitor.h File Reference	3278
7.105	src/main/activemq/state/CommandVisitorAdapter.h File Reference	3278
7.106	src/main/activemq/state/ConnectionState.h File Reference	3279
7.107	src/main/activemq/state/ConnectionStateTracker.h File Reference	3280
7.108	src/main/activemq/state/ConsumerState.h File Reference	3281
7.109	src/main/activemq/state/ProducerState.h File Reference	3281
7.110	src/main/activemq/state/SessionState.h File Reference	3282
7.111	src/main/activemq/state/Tracked.h File Reference	3282
7.112	src/main/activemq/state/TransactionState.h File Reference	3283
7.113	src/main/activemq/threads/CompositeTask.h File Reference	3284
7.114	src/main/activemq/threads/CompositeTaskRunner.h File Reference	3284
7.115	src/main/activemq/threads/DedicatedTaskRunner.h File Reference	3285
7.116	src/main/activemq/threads/Task.h File Reference	3285
7.117	src/main/activemq/threads/TaskRunner.h File Reference	3286
7.118	src/main/activemq/transport/AbstractTransportFactory.h File Reference	3286
7.119	src/main/activemq/transport/CompositeTransport.h File Reference	3287
7.120	src/main/activemq/transport/correlator/FutureResponse.h File Reference	3287
7.121	src/main/activemq/transport/correlator/ResponseCorrelator.h File Reference	3288
7.122	src/main/activemq/transport/DefaultTransportListener.h File Reference	3288
7.123	src/main/activemq/transport/failover/BackupTransport.h File Reference	3289
7.124	src/main/activemq/transport/failover/BackupTransportPool.h File Reference	3289
7.125	src/main/activemq/transport/failover/CloseTransportsTask.h File Reference	3290

7.126src/main/activemq/transport/failover/FailoverTransport.h File Reference	3290
7.127src/main/activemq/transport/failover/FailoverTransportFactory.h File Reference	3291
7.128src/main/activemq/transport/failover/FailoverTransportListener.h File Reference	3292
7.129src/main/activemq/transport/failover/URIPool.h File Reference	3292
7.130src/main/activemq/transport/inactivity/InactivityMonitor.h File Reference	3293
7.131src/main/activemq/transport/inactivity/ReadChecker.h File Reference	3294
7.132src/main/activemq/transport/inactivity/WriteChecker.h File Reference	3294
7.133src/main/activemq/transport/IOTransport.h File Reference	3295
7.134src/main/activemq/transport/logging/LoggingTransport.h File Reference	3295
7.135src/main/activemq/transport/mock/InternalCommandListener.h File Reference	3296
7.136src/main/activemq/transport/mock/MockTransport.h File Reference	3297
7.137src/main/activemq/transport/mock/MockTransportFactory.h File Reference	3297
7.138src/main/activemq/transport/mock/ResponseBuilder.h File Reference	3298
7.139src/main/activemq/transport/tcp/TcpTransport.h File Reference	3298
7.140src/main/activemq/transport/tcp/TcpTransportFactory.h File Reference	3299
7.141src/main/activemq/transport/Transport.h File Reference	3300
7.142src/main/activemq/transport/TransportFactory.h File Reference	3300
7.143src/main/activemq/transport/TransportFilter.h File Reference	3301
7.144src/main/activemq/transport/TransportListener.h File Reference	3302
7.145src/main/activemq/transport/TransportRegistry.h File Reference	3302
7.146src/main/activemq/util/ActiveMQProperties.h File Reference	3303
7.147src/main/activemq/util/CMSExceptionSupport.h File Reference	3303
7.147.1 Define Documentation	3304
7.147.1.1 AMQ_CATCH_ALL_THROW_CMSEXCEPTION	3304
7.148src/main/activemq/util/CompositeData.h File Reference	3305
7.149src/main/activemq/util/Config.h File Reference	3305
7.149.1 Define Documentation	3306
7.149.1.1 AMQCPP_API	3306
7.149.1.2 HAVE_PTHREAD_H	3306
7.149.1.3 HAVE_UUID_T	3306
7.149.1.4 HAVE_UUID_UUID_H	3306
7.150src/main/cms/Config.h File Reference	3306
7.150.1 Define Documentation	3306
7.150.1.1 CMS_API	3306
7.151src/main/decaf/util/Config.h File Reference	3306
7.151.1 Define Documentation	3306

7.151.1.1 DECAF_API	3306
7.151.1.2 DECAF_UNUSED	3306
7.151.1.3 HAVE_PTHREAD_H	3306
7.151.1.4 HAVE_UUID_T	3306
7.151.1.5 HAVE_UUID_UUID_H	3306
7.152src/main/activemq/util/LongSequenceGenerator.h File Reference	3306
7.153src/main/activemq/util/MemoryUsage.h File Reference	3307
7.154src/main/activemq/util/PrimitiveList.h File Reference	3307
7.155src/main/activemq/util/PrimitiveMap.h File Reference	3308
7.156src/main/activemq/util/PrimitiveValueConverter.h File Reference	3309
7.157src/main/activemq/util/PrimitiveValueNode.h File Reference	3309
7.158src/main/activemq/util/URISupport.h File Reference	3310
7.159src/main/activemq/util/Usage.h File Reference	3310
7.160src/main/activemq/wireformat/MarshalAware.h File Reference	3310
7.161src/main/activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h File Reference	3311
7.162src/main/activemq/wireformat/openwire/marshal/DataStreamMarshaller.h File Reference	3312
7.163src/main/activemq/wireformat/openwire/marshal/PrimitiveTypesMarshaller.h File Reference	3312
7.164src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQBlobMessageMarshaller.h File Reference	3313
7.165src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQBlobMessageMarshaller.h File Reference	3314
7.166src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQBlobMessageMarshaller.h File Reference	3314
7.167src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQBlobMessageMarshaller.h File Reference	3315
7.168src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQBlobMessageMarshaller.h File Reference	3316
7.169src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQBytesMessageMarshaller.h File Reference	3316
7.170src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQBytesMessageMarshaller.h File Reference	3317
7.171src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQBytesMessageMarshaller.h File Reference	3318
7.172src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQBytesMessageMarshaller.h File Reference	3318
7.173src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQBytesMessageMarshaller.h File Reference	3319

7.174	src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQDestinationMarshaller.h	
	File Reference	3320
7.175	src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQDestinationMarshaller.h	
	File Reference	3320
7.176	src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQDestinationMarshaller.h	
	File Reference	3321
7.177	src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQDestinationMarshaller.h	
	File Reference	3322
7.178	src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQDestinationMarshaller.h	
	File Reference	3322
7.179	src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQMapMessageMarshaller.h	
	File Reference	3323
7.180	src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQMapMessageMarshaller.h	
	File Reference	3324
7.181	src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQMapMessageMarshaller.h	
	File Reference	3324
7.182	src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQMapMessageMarshaller.h	
	File Reference	3325
7.183	src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQMapMessageMarshaller.h	
	File Reference	3326
7.184	src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQMessageMarshaller.h	
	File Reference	3326
7.185	src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQMessageMarshaller.h	
	File Reference	3327
7.186	src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQMessageMarshaller.h	
	File Reference	3328
7.187	src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQMessageMarshaller.h	
	File Reference	3328
7.188	src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQMessageMarshaller.h	
	File Reference	3329
7.189	src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQObjectMessageMarshaller.h	
	File Reference	3330
7.190	src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQObjectMessageMarshaller.h	
	File Reference	3330
7.191	src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQObjectMessageMarshaller.h	
	File Reference	3331
7.192	src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQObjectMessageMarshaller.h	
	File Reference	3332
7.193	src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQObjectMessageMarshaller.h	
	File Reference	3332
7.194	src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQQueueMarshaller.h	
	File Reference	3333
7.195	src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQQueueMarshaller.h	
	File Reference	3334

7.196	src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQQueueMarshaller.h	
	File Reference	3334
7.197	src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQQueueMarshaller.h	
	File Reference	3335
7.198	src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQQueueMarshaller.h	
	File Reference	3336
7.199	src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQStreamMessageMarshaller.h	
	File Reference	3336
7.200	src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQStreamMessageMarshaller.h	
	File Reference	3337
7.201	src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQStreamMessageMarshaller.h	
	File Reference	3338
7.202	src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQStreamMessageMarshaller.h	
	File Reference	3338
7.203	src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQStreamMessageMarshaller.h	
	File Reference	3339
7.204	src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTempDestinationMarshaller.h	
	File Reference	3340
7.205	src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTempDestinationMarshaller.h	
	File Reference	3340
7.206	src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTempDestinationMarshaller.h	
	File Reference	3341
7.207	src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTempDestinationMarshaller.h	
	File Reference	3342
7.208	src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTempDestinationMarshaller.h	
	File Reference	3342
7.209	src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTempQueueMarshaller.h	
	File Reference	3343
7.210	src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTempQueueMarshaller.h	
	File Reference	3344
7.211	src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTempQueueMarshaller.h	
	File Reference	3344
7.212	src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTempQueueMarshaller.h	
	File Reference	3345
7.213	src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTempQueueMarshaller.h	
	File Reference	3346
7.214	src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTempTopicMarshaller.h	
	File Reference	3346
7.215	src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTempTopicMarshaller.h	
	File Reference	3347
7.216	src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTempTopicMarshaller.h	
	File Reference	3348
7.217	src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTempTopicMarshaller.h	
	File Reference	3348

7.218	src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTempTopicMarshaller.h	
	File Reference	3349
7.219	src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTextMessageMarshaller.h	
	File Reference	3350
7.220	src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTextMessageMarshaller.h	
	File Reference	3350
7.221	src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTextMessageMarshaller.h	
	File Reference	3351
7.222	src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTextMessageMarshaller.h	
	File Reference	3352
7.223	src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTextMessageMarshaller.h	
	File Reference	3352
7.224	src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTopicMarshaller.h	
	File Reference	3353
7.225	src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTopicMarshaller.h	
	File Reference	3354
7.226	src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTopicMarshaller.h	
	File Reference	3354
7.227	src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTopicMarshaller.h	
	File Reference	3355
7.228	src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTopicMarshaller.h	
	File Reference	3356
7.229	src/main/activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h	
	File Reference	3356
7.230	src/main/activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h	
	File Reference	3357
7.231	src/main/activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h	
	File Reference	3358
7.232	src/main/activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h	
	File Reference	3358
7.233	src/main/activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h	
	File Reference	3359
7.234	src/main/activemq/wireformat/openwire/marshal/v1/BrokerIdMarshaller.h	File
	Reference	3360
7.235	src/main/activemq/wireformat/openwire/marshal/v2/BrokerIdMarshaller.h	File
	Reference	3360
7.236	src/main/activemq/wireformat/openwire/marshal/v3/BrokerIdMarshaller.h	File
	Reference	3361
7.237	src/main/activemq/wireformat/openwire/marshal/v4/BrokerIdMarshaller.h	File
	Reference	3362
7.238	src/main/activemq/wireformat/openwire/marshal/v5/BrokerIdMarshaller.h	File
	Reference	3362
7.239	src/main/activemq/wireformat/openwire/marshal/v1/BrokerInfoMarshaller.h	File
	Reference	3363

7.240src/main/activemq/wireformat/openwire/marshal/v2/BrokerInfoMarshaller.h File	
Reference	3364
7.241src/main/activemq/wireformat/openwire/marshal/v3/BrokerInfoMarshaller.h File	
Reference	3364
7.242src/main/activemq/wireformat/openwire/marshal/v4/BrokerInfoMarshaller.h File	
Reference	3365
7.243src/main/activemq/wireformat/openwire/marshal/v5/BrokerInfoMarshaller.h File	
Reference	3366
7.244src/main/activemq/wireformat/openwire/marshal/v1/ConnectionControlMarshaller.h	
File Reference	3366
7.245src/main/activemq/wireformat/openwire/marshal/v2/ConnectionControlMarshaller.h	
File Reference	3367
7.246src/main/activemq/wireformat/openwire/marshal/v3/ConnectionControlMarshaller.h	
File Reference	3368
7.247src/main/activemq/wireformat/openwire/marshal/v4/ConnectionControlMarshaller.h	
File Reference	3368
7.248src/main/activemq/wireformat/openwire/marshal/v5/ConnectionControlMarshaller.h	
File Reference	3369
7.249src/main/activemq/wireformat/openwire/marshal/v1/ConnectionErrorMarshaller.h	
File Reference	3370
7.250src/main/activemq/wireformat/openwire/marshal/v2/ConnectionErrorMarshaller.h	
File Reference	3370
7.251src/main/activemq/wireformat/openwire/marshal/v3/ConnectionErrorMarshaller.h	
File Reference	3371
7.252src/main/activemq/wireformat/openwire/marshal/v4/ConnectionErrorMarshaller.h	
File Reference	3372
7.253src/main/activemq/wireformat/openwire/marshal/v5/ConnectionErrorMarshaller.h	
File Reference	3372
7.254src/main/activemq/wireformat/openwire/marshal/v1/ConnectionIdMarshaller.h	
File Reference	3373
7.255src/main/activemq/wireformat/openwire/marshal/v2/ConnectionIdMarshaller.h	
File Reference	3374
7.256src/main/activemq/wireformat/openwire/marshal/v3/ConnectionIdMarshaller.h	
File Reference	3374
7.257src/main/activemq/wireformat/openwire/marshal/v4/ConnectionIdMarshaller.h	
File Reference	3375
7.258src/main/activemq/wireformat/openwire/marshal/v5/ConnectionIdMarshaller.h	
File Reference	3376
7.259src/main/activemq/wireformat/openwire/marshal/v1/ConnectionInfoMarshaller.h	
File Reference	3376
7.260src/main/activemq/wireformat/openwire/marshal/v2/ConnectionInfoMarshaller.h	
File Reference	3377
7.261src/main/activemq/wireformat/openwire/marshal/v3/ConnectionInfoMarshaller.h	
File Reference	3378

7.262	src/main/activemq/wireformat/openwire/marshal/v4/ConnectionInfoMarshaller.h	
	File Reference	3378
7.263	src/main/activemq/wireformat/openwire/marshal/v5/ConnectionInfoMarshaller.h	
	File Reference	3379
7.264	src/main/activemq/wireformat/openwire/marshal/v1/ConsumerControlMarshaller.h	
	File Reference	3380
7.265	src/main/activemq/wireformat/openwire/marshal/v2/ConsumerControlMarshaller.h	
	File Reference	3380
7.266	src/main/activemq/wireformat/openwire/marshal/v3/ConsumerControlMarshaller.h	
	File Reference	3381
7.267	src/main/activemq/wireformat/openwire/marshal/v4/ConsumerControlMarshaller.h	
	File Reference	3382
7.268	src/main/activemq/wireformat/openwire/marshal/v5/ConsumerControlMarshaller.h	
	File Reference	3382
7.269	src/main/activemq/wireformat/openwire/marshal/v1/ConsumerIdMarshaller.h	
	File Reference	3383
7.270	src/main/activemq/wireformat/openwire/marshal/v2/ConsumerIdMarshaller.h	
	File Reference	3384
7.271	src/main/activemq/wireformat/openwire/marshal/v3/ConsumerIdMarshaller.h	
	File Reference	3384
7.272	src/main/activemq/wireformat/openwire/marshal/v4/ConsumerIdMarshaller.h	
	File Reference	3385
7.273	src/main/activemq/wireformat/openwire/marshal/v5/ConsumerIdMarshaller.h	
	File Reference	3386
7.274	src/main/activemq/wireformat/openwire/marshal/v1/ConsumerInfoMarshaller.h	
	File Reference	3386
7.275	src/main/activemq/wireformat/openwire/marshal/v2/ConsumerInfoMarshaller.h	
	File Reference	3387
7.276	src/main/activemq/wireformat/openwire/marshal/v3/ConsumerInfoMarshaller.h	
	File Reference	3388
7.277	src/main/activemq/wireformat/openwire/marshal/v4/ConsumerInfoMarshaller.h	
	File Reference	3388
7.278	src/main/activemq/wireformat/openwire/marshal/v5/ConsumerInfoMarshaller.h	
	File Reference	3389
7.279	src/main/activemq/wireformat/openwire/marshal/v1/ControlCommandMarshaller.h	
	File Reference	3390
7.280	src/main/activemq/wireformat/openwire/marshal/v2/ControlCommandMarshaller.h	
	File Reference	3390
7.281	src/main/activemq/wireformat/openwire/marshal/v3/ControlCommandMarshaller.h	
	File Reference	3391
7.282	src/main/activemq/wireformat/openwire/marshal/v4/ControlCommandMarshaller.h	
	File Reference	3392
7.283	src/main/activemq/wireformat/openwire/marshal/v5/ControlCommandMarshaller.h	
	File Reference	3392

7.284	src/main/activemq/wireformat/openwire/marshal/v1/DataArrayResponseMarshaller.h	
	File Reference	3393
7.285	src/main/activemq/wireformat/openwire/marshal/v2/DataArrayResponseMarshaller.h	
	File Reference	3394
7.286	src/main/activemq/wireformat/openwire/marshal/v3/DataArrayResponseMarshaller.h	
	File Reference	3394
7.287	src/main/activemq/wireformat/openwire/marshal/v4/DataArrayResponseMarshaller.h	
	File Reference	3395
7.288	src/main/activemq/wireformat/openwire/marshal/v5/DataArrayResponseMarshaller.h	
	File Reference	3396
7.289	src/main/activemq/wireformat/openwire/marshal/v1/DataResponseMarshaller.h	
	File Reference	3396
7.290	src/main/activemq/wireformat/openwire/marshal/v2/DataResponseMarshaller.h	
	File Reference	3397
7.291	src/main/activemq/wireformat/openwire/marshal/v3/DataResponseMarshaller.h	
	File Reference	3398
7.292	src/main/activemq/wireformat/openwire/marshal/v4/DataResponseMarshaller.h	
	File Reference	3398
7.293	src/main/activemq/wireformat/openwire/marshal/v5/DataResponseMarshaller.h	
	File Reference	3399
7.294	src/main/activemq/wireformat/openwire/marshal/v1/DestinationInfoMarshaller.h	
	File Reference	3400
7.295	src/main/activemq/wireformat/openwire/marshal/v2/DestinationInfoMarshaller.h	
	File Reference	3400
7.296	src/main/activemq/wireformat/openwire/marshal/v3/DestinationInfoMarshaller.h	
	File Reference	3401
7.297	src/main/activemq/wireformat/openwire/marshal/v4/DestinationInfoMarshaller.h	
	File Reference	3402
7.298	src/main/activemq/wireformat/openwire/marshal/v5/DestinationInfoMarshaller.h	
	File Reference	3402
7.299	src/main/activemq/wireformat/openwire/marshal/v1/DiscoveryEventMarshaller.h	
	File Reference	3403
7.300	src/main/activemq/wireformat/openwire/marshal/v2/DiscoveryEventMarshaller.h	
	File Reference	3404
7.301	src/main/activemq/wireformat/openwire/marshal/v3/DiscoveryEventMarshaller.h	
	File Reference	3404
7.302	src/main/activemq/wireformat/openwire/marshal/v4/DiscoveryEventMarshaller.h	
	File Reference	3405
7.303	src/main/activemq/wireformat/openwire/marshal/v5/DiscoveryEventMarshaller.h	
	File Reference	3406
7.304	src/main/activemq/wireformat/openwire/marshal/v1/ExceptionResponseMarshaller.h	
	File Reference	3406
7.305	src/main/activemq/wireformat/openwire/marshal/v2/ExceptionResponseMarshaller.h	
	File Reference	3407

7.306	src/main/activemq/wireformat/openwire/marshal/v3/ExceptionResponseMarshaller.h	
	File Reference	3408
7.307	src/main/activemq/wireformat/openwire/marshal/v4/ExceptionResponseMarshaller.h	
	File Reference	3408
7.308	src/main/activemq/wireformat/openwire/marshal/v5/ExceptionResponseMarshaller.h	
	File Reference	3409
7.309	src/main/activemq/wireformat/openwire/marshal/v1/FlushCommandMarshaller.h	
	File Reference	3410
7.310	src/main/activemq/wireformat/openwire/marshal/v2/FlushCommandMarshaller.h	
	File Reference	3410
7.311	src/main/activemq/wireformat/openwire/marshal/v3/FlushCommandMarshaller.h	
	File Reference	3411
7.312	src/main/activemq/wireformat/openwire/marshal/v4/FlushCommandMarshaller.h	
	File Reference	3412
7.313	src/main/activemq/wireformat/openwire/marshal/v5/FlushCommandMarshaller.h	
	File Reference	3412
7.314	src/main/activemq/wireformat/openwire/marshal/v1/IntegerResponseMarshaller.h	
	File Reference	3413
7.315	src/main/activemq/wireformat/openwire/marshal/v2/IntegerResponseMarshaller.h	
	File Reference	3414
7.316	src/main/activemq/wireformat/openwire/marshal/v3/IntegerResponseMarshaller.h	
	File Reference	3414
7.317	src/main/activemq/wireformat/openwire/marshal/v4/IntegerResponseMarshaller.h	
	File Reference	3415
7.318	src/main/activemq/wireformat/openwire/marshal/v5/IntegerResponseMarshaller.h	
	File Reference	3416
7.319	src/main/activemq/wireformat/openwire/marshal/v1/JournalQueueAckMarshaller.h	
	File Reference	3416
7.320	src/main/activemq/wireformat/openwire/marshal/v2/JournalQueueAckMarshaller.h	
	File Reference	3417
7.321	src/main/activemq/wireformat/openwire/marshal/v3/JournalQueueAckMarshaller.h	
	File Reference	3418
7.322	src/main/activemq/wireformat/openwire/marshal/v4/JournalQueueAckMarshaller.h	
	File Reference	3418
7.323	src/main/activemq/wireformat/openwire/marshal/v5/JournalQueueAckMarshaller.h	
	File Reference	3419
7.324	src/main/activemq/wireformat/openwire/marshal/v1/JournalTopicAckMarshaller.h	
	File Reference	3420
7.325	src/main/activemq/wireformat/openwire/marshal/v2/JournalTopicAckMarshaller.h	
	File Reference	3420
7.326	src/main/activemq/wireformat/openwire/marshal/v3/JournalTopicAckMarshaller.h	
	File Reference	3421
7.327	src/main/activemq/wireformat/openwire/marshal/v4/JournalTopicAckMarshaller.h	
	File Reference	3422

7.328	src/main/activemq/wireformat/openwire/marshal/v5/JournalTopicAckMarshaller.h	
	File Reference	3422
7.329	src/main/activemq/wireformat/openwire/marshal/v1/JournalTraceMarshaller.h	
	File Reference	3423
7.330	src/main/activemq/wireformat/openwire/marshal/v2/JournalTraceMarshaller.h	
	File Reference	3424
7.331	src/main/activemq/wireformat/openwire/marshal/v3/JournalTraceMarshaller.h	
	File Reference	3424
7.332	src/main/activemq/wireformat/openwire/marshal/v4/JournalTraceMarshaller.h	
	File Reference	3425
7.333	src/main/activemq/wireformat/openwire/marshal/v5/JournalTraceMarshaller.h	
	File Reference	3426
7.334	src/main/activemq/wireformat/openwire/marshal/v1/JournalTransactionMarshaller.h	
	File Reference	3426
7.335	src/main/activemq/wireformat/openwire/marshal/v2/JournalTransactionMarshaller.h	
	File Reference	3427
7.336	src/main/activemq/wireformat/openwire/marshal/v3/JournalTransactionMarshaller.h	
	File Reference	3428
7.337	src/main/activemq/wireformat/openwire/marshal/v4/JournalTransactionMarshaller.h	
	File Reference	3428
7.338	src/main/activemq/wireformat/openwire/marshal/v5/JournalTransactionMarshaller.h	
	File Reference	3429
7.339	src/main/activemq/wireformat/openwire/marshal/v1/KeepAliveInfoMarshaller.h	
	File Reference	3430
7.340	src/main/activemq/wireformat/openwire/marshal/v2/KeepAliveInfoMarshaller.h	
	File Reference	3430
7.341	src/main/activemq/wireformat/openwire/marshal/v3/KeepAliveInfoMarshaller.h	
	File Reference	3431
7.342	src/main/activemq/wireformat/openwire/marshal/v4/KeepAliveInfoMarshaller.h	
	File Reference	3432
7.343	src/main/activemq/wireformat/openwire/marshal/v5/KeepAliveInfoMarshaller.h	
	File Reference	3432
7.344	src/main/activemq/wireformat/openwire/marshal/v1/LastPartialCommandMarshaller.h	
	File Reference	3433
7.345	src/main/activemq/wireformat/openwire/marshal/v2/LastPartialCommandMarshaller.h	
	File Reference	3434
7.346	src/main/activemq/wireformat/openwire/marshal/v3/LastPartialCommandMarshaller.h	
	File Reference	3434
7.347	src/main/activemq/wireformat/openwire/marshal/v4/LastPartialCommandMarshaller.h	
	File Reference	3435
7.348	src/main/activemq/wireformat/openwire/marshal/v5/LastPartialCommandMarshaller.h	
	File Reference	3436
7.349	src/main/activemq/wireformat/openwire/marshal/v1/LocalTransactionIdMarshaller.h	
	File Reference	3436

7.350	src/main/activemq/wireformat/openwire/marshal/v2/LocalTransactionIdMarshaller.h	
	File Reference	3437
7.351	src/main/activemq/wireformat/openwire/marshal/v3/LocalTransactionIdMarshaller.h	
	File Reference	3438
7.352	src/main/activemq/wireformat/openwire/marshal/v4/LocalTransactionIdMarshaller.h	
	File Reference	3438
7.353	src/main/activemq/wireformat/openwire/marshal/v5/LocalTransactionIdMarshaller.h	
	File Reference	3439
7.354	src/main/activemq/wireformat/openwire/marshal/v1/MarshallerFactory.h	File
	Reference	3440
7.355	src/main/activemq/wireformat/openwire/marshal/v2/MarshallerFactory.h	File
	Reference	3440
7.356	src/main/activemq/wireformat/openwire/marshal/v3/MarshallerFactory.h	File
	Reference	3441
7.357	src/main/activemq/wireformat/openwire/marshal/v4/MarshallerFactory.h	File
	Reference	3441
7.358	src/main/activemq/wireformat/openwire/marshal/v5/MarshallerFactory.h	File
	Reference	3442
7.359	src/main/activemq/wireformat/openwire/marshal/v1/MessageAckMarshaller.h	
	File Reference	3442
7.360	src/main/activemq/wireformat/openwire/marshal/v2/MessageAckMarshaller.h	
	File Reference	3443
7.361	src/main/activemq/wireformat/openwire/marshal/v3/MessageAckMarshaller.h	
	File Reference	3443
7.362	src/main/activemq/wireformat/openwire/marshal/v4/MessageAckMarshaller.h	
	File Reference	3444
7.363	src/main/activemq/wireformat/openwire/marshal/v5/MessageAckMarshaller.h	
	File Reference	3445
7.364	src/main/activemq/wireformat/openwire/marshal/v1/MessageDispatchMarshaller.h	
	File Reference	3445
7.365	src/main/activemq/wireformat/openwire/marshal/v2/MessageDispatchMarshaller.h	
	File Reference	3446
7.366	src/main/activemq/wireformat/openwire/marshal/v3/MessageDispatchMarshaller.h	
	File Reference	3447
7.367	src/main/activemq/wireformat/openwire/marshal/v4/MessageDispatchMarshaller.h	
	File Reference	3447
7.368	src/main/activemq/wireformat/openwire/marshal/v5/MessageDispatchMarshaller.h	
	File Reference	3448
7.369	src/main/activemq/wireformat/openwire/marshal/v1/MessageDispatchNotificationMarshaller.h	
	File Reference	3449
7.370	src/main/activemq/wireformat/openwire/marshal/v2/MessageDispatchNotificationMarshaller.h	
	File Reference	3450
7.371	src/main/activemq/wireformat/openwire/marshal/v3/MessageDispatchNotificationMarshaller.h	
	File Reference	3450

7.372	src/main/activemq/wireformat/openwire/marshal/v4/MessageDispatchNotificationMarshaller.h	
	File Reference	3451
7.373	src/main/activemq/wireformat/openwire/marshal/v5/MessageDispatchNotificationMarshaller.h	
	File Reference	3452
7.374	src/main/activemq/wireformat/openwire/marshal/v1/MessageIdMarshaller.h	File
	Reference	3452
7.375	src/main/activemq/wireformat/openwire/marshal/v2/MessageIdMarshaller.h	File
	Reference	3453
7.376	src/main/activemq/wireformat/openwire/marshal/v3/MessageIdMarshaller.h	File
	Reference	3454
7.377	src/main/activemq/wireformat/openwire/marshal/v4/MessageIdMarshaller.h	File
	Reference	3454
7.378	src/main/activemq/wireformat/openwire/marshal/v5/MessageIdMarshaller.h	File
	Reference	3455
7.379	src/main/activemq/wireformat/openwire/marshal/v1/MessageMarshaller.h	File
	Reference	3456
7.380	src/main/activemq/wireformat/openwire/marshal/v2/MessageMarshaller.h	File
	Reference	3456
7.381	src/main/activemq/wireformat/openwire/marshal/v3/MessageMarshaller.h	File
	Reference	3457
7.382	src/main/activemq/wireformat/openwire/marshal/v4/MessageMarshaller.h	File
	Reference	3458
7.383	src/main/activemq/wireformat/openwire/marshal/v5/MessageMarshaller.h	File
	Reference	3458
7.384	src/main/activemq/wireformat/openwire/marshal/v1/MessagePullMarshaller.h	
	File Reference	3459
7.385	src/main/activemq/wireformat/openwire/marshal/v2/MessagePullMarshaller.h	
	File Reference	3460
7.386	src/main/activemq/wireformat/openwire/marshal/v3/MessagePullMarshaller.h	
	File Reference	3460
7.387	src/main/activemq/wireformat/openwire/marshal/v4/MessagePullMarshaller.h	
	File Reference	3461
7.388	src/main/activemq/wireformat/openwire/marshal/v5/MessagePullMarshaller.h	
	File Reference	3462
7.389	src/main/activemq/wireformat/openwire/marshal/v1/NetworkBridgeFilterMarshaller.h	
	File Reference	3462
7.390	src/main/activemq/wireformat/openwire/marshal/v2/NetworkBridgeFilterMarshaller.h	
	File Reference	3463
7.391	src/main/activemq/wireformat/openwire/marshal/v3/NetworkBridgeFilterMarshaller.h	
	File Reference	3464
7.392	src/main/activemq/wireformat/openwire/marshal/v4/NetworkBridgeFilterMarshaller.h	
	File Reference	3464
7.393	src/main/activemq/wireformat/openwire/marshal/v5/NetworkBridgeFilterMarshaller.h	
	File Reference	3465

7.394	src/main/activemq/wireformat/openwire/marshal/v1/PartialCommandMarshaller.h	
	File Reference	3466
7.395	src/main/activemq/wireformat/openwire/marshal/v2/PartialCommandMarshaller.h	
	File Reference	3466
7.396	src/main/activemq/wireformat/openwire/marshal/v3/PartialCommandMarshaller.h	
	File Reference	3467
7.397	src/main/activemq/wireformat/openwire/marshal/v4/PartialCommandMarshaller.h	
	File Reference	3468
7.398	src/main/activemq/wireformat/openwire/marshal/v5/PartialCommandMarshaller.h	
	File Reference	3468
7.399	src/main/activemq/wireformat/openwire/marshal/v1/ProducerAckMarshaller.h	
	File Reference	3469
7.400	src/main/activemq/wireformat/openwire/marshal/v2/ProducerAckMarshaller.h	
	File Reference	3470
7.401	src/main/activemq/wireformat/openwire/marshal/v3/ProducerAckMarshaller.h	
	File Reference	3470
7.402	src/main/activemq/wireformat/openwire/marshal/v4/ProducerAckMarshaller.h	
	File Reference	3471
7.403	src/main/activemq/wireformat/openwire/marshal/v5/ProducerAckMarshaller.h	
	File Reference	3472
7.404	src/main/activemq/wireformat/openwire/marshal/v1/ProducerIdMarshaller.h	
	File Reference	3472
7.405	src/main/activemq/wireformat/openwire/marshal/v2/ProducerIdMarshaller.h	
	File Reference	3473
7.406	src/main/activemq/wireformat/openwire/marshal/v3/ProducerIdMarshaller.h	
	File Reference	3474
7.407	src/main/activemq/wireformat/openwire/marshal/v4/ProducerIdMarshaller.h	
	File Reference	3474
7.408	src/main/activemq/wireformat/openwire/marshal/v5/ProducerIdMarshaller.h	
	File Reference	3475
7.409	src/main/activemq/wireformat/openwire/marshal/v1/ProducerInfoMarshaller.h	
	File Reference	3476
7.410	src/main/activemq/wireformat/openwire/marshal/v2/ProducerInfoMarshaller.h	
	File Reference	3476
7.411	src/main/activemq/wireformat/openwire/marshal/v3/ProducerInfoMarshaller.h	
	File Reference	3477
7.412	src/main/activemq/wireformat/openwire/marshal/v4/ProducerInfoMarshaller.h	
	File Reference	3478
7.413	src/main/activemq/wireformat/openwire/marshal/v5/ProducerInfoMarshaller.h	
	File Reference	3478
7.414	src/main/activemq/wireformat/openwire/marshal/v1/RemoveInfoMarshaller.h	
	File Reference	3479
7.415	src/main/activemq/wireformat/openwire/marshal/v2/RemoveInfoMarshaller.h	
	File Reference	3480

7.416	src/main/activemq/wireformat/openwire/marshal/v3/RemoveInfoMarshaller.h	
	File Reference	3480
7.417	src/main/activemq/wireformat/openwire/marshal/v4/RemoveInfoMarshaller.h	
	File Reference	3481
7.418	src/main/activemq/wireformat/openwire/marshal/v5/RemoveInfoMarshaller.h	
	File Reference	3482
7.419	src/main/activemq/wireformat/openwire/marshal/v1/RemoveSubscriptionInfoMarshaller.h	
	File Reference	3482
7.420	src/main/activemq/wireformat/openwire/marshal/v2/RemoveSubscriptionInfoMarshaller.h	
	File Reference	3483
7.421	src/main/activemq/wireformat/openwire/marshal/v3/RemoveSubscriptionInfoMarshaller.h	
	File Reference	3484
7.422	src/main/activemq/wireformat/openwire/marshal/v4/RemoveSubscriptionInfoMarshaller.h	
	File Reference	3484
7.423	src/main/activemq/wireformat/openwire/marshal/v5/RemoveSubscriptionInfoMarshaller.h	
	File Reference	3485
7.424	src/main/activemq/wireformat/openwire/marshal/v1/ReplayCommandMarshaller.h	
	File Reference	3486
7.425	src/main/activemq/wireformat/openwire/marshal/v2/ReplayCommandMarshaller.h	
	File Reference	3486
7.426	src/main/activemq/wireformat/openwire/marshal/v3/ReplayCommandMarshaller.h	
	File Reference	3487
7.427	src/main/activemq/wireformat/openwire/marshal/v4/ReplayCommandMarshaller.h	
	File Reference	3488
7.428	src/main/activemq/wireformat/openwire/marshal/v5/ReplayCommandMarshaller.h	
	File Reference	3488
7.429	src/main/activemq/wireformat/openwire/marshal/v1/ResponseMarshaller.h	File
	Reference	3489
7.430	src/main/activemq/wireformat/openwire/marshal/v2/ResponseMarshaller.h	File
	Reference	3490
7.431	src/main/activemq/wireformat/openwire/marshal/v3/ResponseMarshaller.h	File
	Reference	3490
7.432	src/main/activemq/wireformat/openwire/marshal/v4/ResponseMarshaller.h	File
	Reference	3491
7.433	src/main/activemq/wireformat/openwire/marshal/v5/ResponseMarshaller.h	File
	Reference	3492
7.434	src/main/activemq/wireformat/openwire/marshal/v1/SessionIdMarshaller.h	File
	Reference	3492
7.435	src/main/activemq/wireformat/openwire/marshal/v2/SessionIdMarshaller.h	File
	Reference	3493
7.436	src/main/activemq/wireformat/openwire/marshal/v3/SessionIdMarshaller.h	File
	Reference	3494
7.437	src/main/activemq/wireformat/openwire/marshal/v4/SessionIdMarshaller.h	File
	Reference	3494

7.438src/main/activemq/wireformat/openwire/marshal/v5/SessionIdMarshaller.h	File
Reference	3495
7.439src/main/activemq/wireformat/openwire/marshal/v1/SessionInfoMarshaller.h	
File Reference	3496
7.440src/main/activemq/wireformat/openwire/marshal/v2/SessionInfoMarshaller.h	
File Reference	3496
7.441src/main/activemq/wireformat/openwire/marshal/v3/SessionInfoMarshaller.h	
File Reference	3497
7.442src/main/activemq/wireformat/openwire/marshal/v4/SessionInfoMarshaller.h	
File Reference	3498
7.443src/main/activemq/wireformat/openwire/marshal/v5/SessionInfoMarshaller.h	
File Reference	3498
7.444src/main/activemq/wireformat/openwire/marshal/v1/ShutdownInfoMarshaller.h	
File Reference	3499
7.445src/main/activemq/wireformat/openwire/marshal/v2/ShutdownInfoMarshaller.h	
File Reference	3500
7.446src/main/activemq/wireformat/openwire/marshal/v3/ShutdownInfoMarshaller.h	
File Reference	3500
7.447src/main/activemq/wireformat/openwire/marshal/v4/ShutdownInfoMarshaller.h	
File Reference	3501
7.448src/main/activemq/wireformat/openwire/marshal/v5/ShutdownInfoMarshaller.h	
File Reference	3502
7.449src/main/activemq/wireformat/openwire/marshal/v1/SubscriptionInfoMarshaller.h	
File Reference	3502
7.450src/main/activemq/wireformat/openwire/marshal/v2/SubscriptionInfoMarshaller.h	
File Reference	3503
7.451src/main/activemq/wireformat/openwire/marshal/v3/SubscriptionInfoMarshaller.h	
File Reference	3504
7.452src/main/activemq/wireformat/openwire/marshal/v4/SubscriptionInfoMarshaller.h	
File Reference	3504
7.453src/main/activemq/wireformat/openwire/marshal/v5/SubscriptionInfoMarshaller.h	
File Reference	3505
7.454src/main/activemq/wireformat/openwire/marshal/v1/TransactionIdMarshaller.h	
File Reference	3506
7.455src/main/activemq/wireformat/openwire/marshal/v2/TransactionIdMarshaller.h	
File Reference	3506
7.456src/main/activemq/wireformat/openwire/marshal/v3/TransactionIdMarshaller.h	
File Reference	3507
7.457src/main/activemq/wireformat/openwire/marshal/v4/TransactionIdMarshaller.h	
File Reference	3508
7.458src/main/activemq/wireformat/openwire/marshal/v5/TransactionIdMarshaller.h	
File Reference	3508
7.459src/main/activemq/wireformat/openwire/marshal/v1/TransactionInfoMarshaller.h	
File Reference	3509

7.460src/main/activemq/wireformat/openwire/marshal/v2/TransactionInfoMarshaller.h File Reference	3510
7.461src/main/activemq/wireformat/openwire/marshal/v3/TransactionInfoMarshaller.h File Reference	3510
7.462src/main/activemq/wireformat/openwire/marshal/v4/TransactionInfoMarshaller.h File Reference	3511
7.463src/main/activemq/wireformat/openwire/marshal/v5/TransactionInfoMarshaller.h File Reference	3512
7.464src/main/activemq/wireformat/openwire/marshal/v1/WireFormatInfoMarshaller.h File Reference	3512
7.465src/main/activemq/wireformat/openwire/marshal/v2/WireFormatInfoMarshaller.h File Reference	3513
7.466src/main/activemq/wireformat/openwire/marshal/v3/WireFormatInfoMarshaller.h File Reference	3514
7.467src/main/activemq/wireformat/openwire/marshal/v4/WireFormatInfoMarshaller.h File Reference	3514
7.468src/main/activemq/wireformat/openwire/marshal/v5/WireFormatInfoMarshaller.h File Reference	3515
7.469src/main/activemq/wireformat/openwire/marshal/v1/XATransactionIdMarshaller.h File Reference	3516
7.470src/main/activemq/wireformat/openwire/marshal/v2/XATransactionIdMarshaller.h File Reference	3516
7.471src/main/activemq/wireformat/openwire/marshal/v3/XATransactionIdMarshaller.h File Reference	3517
7.472src/main/activemq/wireformat/openwire/marshal/v4/XATransactionIdMarshaller.h File Reference	3518
7.473src/main/activemq/wireformat/openwire/marshal/v5/XATransactionIdMarshaller.h File Reference	3518
7.474src/main/activemq/wireformat/openwire/OpenWireFormat.h File Reference	3519
7.475src/main/activemq/wireformat/openwire/OpenWireFormatFactory.h File Reference	3520
7.476src/main/activemq/wireformat/openwire/OpenWireFormatNegotiator.h File Ref- erence	3520
7.477src/main/activemq/wireformat/openwire/OpenWireResponseBuilder.h File Refer- ence	3521
7.478src/main/activemq/wireformat/openwire/utils/BooleanStream.h File Reference . .	3521
7.479src/main/activemq/wireformat/openwire/utils/HexTable.h File Reference	3522
7.480src/main/activemq/wireformat/openwire/utils/MessagePropertyInterceptor.h File Reference	3523
7.481src/main/activemq/wireformat/openwire/utils/OpenwireStringSupport.h File Ref- erence	3523
7.482src/main/activemq/wireformat/stomp/StompCommandConstants.h File Reference	3524
7.483src/main/activemq/wireformat/stomp/StompFrame.h File Reference	3524

7.484src/main/activemq/wireformat/stomp/StompHelper.h File Reference	3525
7.485src/main/activemq/wireformat/stomp/StompWireFormat.h File Reference	3525
7.486src/main/activemq/wireformat/stomp/StompWireFormatFactory.h File Reference	3526
7.487src/main/activemq/wireformat/WireFormat.h File Reference	3527
7.488src/main/activemq/wireformat/WireFormatFactory.h File Reference	3527
7.489src/main/activemq/wireformat/WireFormatNegotiator.h File Reference	3528
7.490src/main/activemq/wireformat/WireFormatRegistry.h File Reference	3528
7.491src/main/cms/BytesMessage.h File Reference	3529
7.492src/main/cms/Closeable.h File Reference	3529
7.493src/main/decaf/io/Closeable.h File Reference	3530
7.494src/main/cms/CMSException.h File Reference	3530
7.495src/main/cms/CMSProperties.h File Reference	3531
7.496src/main/cms/CMSSecurityException.h File Reference	3531
7.497src/main/cms/Connection.h File Reference	3532
7.498src/main/cms/ConnectionFactory.h File Reference	3532
7.499src/main/cms/ConnectionMetaData.h File Reference	3533
7.500src/main/cms/DeliveryMode.h File Reference	3533
7.501src/main/cms/Destination.h File Reference	3533
7.502src/main/cms/ExceptionListener.h File Reference	3534
7.503src/main/cms/IllegalStateException.h File Reference	3534
7.504src/main/decaf/lang/exceptions/IllegalStateException.h File Reference	3535
7.505src/main/cms/InvalidClientIdException.h File Reference	3535
7.506src/main/cms/InvalidDestinationException.h File Reference	3536
7.507src/main/cms/InvalidSelectorException.h File Reference	3536
7.508src/main/cms/MapMessage.h File Reference	3536
7.509src/main/cms/MessageConsumer.h File Reference	3537
7.510src/main/cms/MessageEOFException.h File Reference	3537
7.511src/main/cms/MessageFormatException.h File Reference	3538
7.512src/main/cms/MessageListener.h File Reference	3538
7.513src/main/cms/MessageNotReadableException.h File Reference	3539
7.514src/main/cms/MessageNotWriteableException.h File Reference	3539
7.515src/main/cms/MessageProducer.h File Reference	3540
7.516src/main/cms/ObjectMessage.h File Reference	3540
7.517src/main/cms/Queue.h File Reference	3541
7.518src/main/decaf/util/Queue.h File Reference	3541
7.519src/main/cms/QueueBrowser.h File Reference	3542

7.520src/main/cms/Session.h File Reference	3542
7.521src/main/cms/Startable.h File Reference	3543
7.522src/main/cms/Stoppable.h File Reference	3543
7.523src/main/cms/StreamMessage.h File Reference	3544
7.524src/main/cms/TemporaryQueue.h File Reference	3544
7.525src/main/cms/TemporaryTopic.h File Reference	3545
7.526src/main/cms/TextMessage.h File Reference	3545
7.527src/main/cms/Topic.h File Reference	3546
7.528src/main/cms/UnsupportedOperationException.h File Reference	3546
7.529src/main/decaf/lang/exceptions/UnsupportedOperationException.h File Reference	3546
7.530src/main/decaf/internal/AprPool.h File Reference	3547
7.531src/main/decaf/internal/DecafRuntime.h File Reference	3547
7.532src/main/decaf/internal/io/StandardErrorOutputStream.h File Reference	3548
7.533src/main/decaf/internal/io/StandardInputStream.h File Reference	3548
7.534src/main/decaf/internal/io/StandardOutputStream.h File Reference	3549
7.535src/main/decaf/internal/net/URLEncoderDecoder.h File Reference	3549
7.536src/main/decaf/internal/net/URIHelper.h File Reference	3550
7.537src/main/decaf/internal/net/URIType.h File Reference	3550
7.538src/main/decaf/internal/nio/BufferFactory.h File Reference	3551
7.539src/main/decaf/internal/nio/ByteBuffer.h File Reference	3551
7.540src/main/decaf/internal/nio/ByteBufferPerspective.h File Reference	3552
7.541src/main/decaf/internal/nio/CharArrayBuffer.h File Reference	3552
7.542src/main/decaf/internal/nio/DoubleArrayBuffer.h File Reference	3553
7.543src/main/decaf/internal/nio/FloatArrayBuffer.h File Reference	3554
7.544src/main/decaf/internal/nio/IntArrayBuffer.h File Reference	3554
7.545src/main/decaf/internal/nio/LongArrayBuffer.h File Reference	3555
7.546src/main/decaf/internal/nio/ShortArrayBuffer.h File Reference	3555
7.547src/main/decaf/internal/util/ByteArrayAdapter.h File Reference	3556
7.548src/main/decaf/internal/util/concurrent/ConditionImpl.h File Reference	3556
7.549src/main/decaf/internal/util/concurrent/MutexImpl.h File Reference	3557
7.550src/main/decaf/internal/util/concurrent/SynchronizableImpl.h File Reference	3557
7.551src/main/decaf/internal/util/concurrent/Transferer.h File Reference	3558
7.552src/main/decaf/internal/util/concurrent/TransferQueue.h File Reference	3558
7.553src/main/decaf/internal/util/concurrent/TransferStack.h File Reference	3559
7.554src/main/decaf/internal/util/concurrent/unix/ConditionHandle.h File Reference	3559
7.555src/main/decaf/internal/util/concurrent/windows/ConditionHandle.h File Reference	3560

7.556	src/main/decaf/internal/util/concurrent/unix/MutexHandle.h File Reference . . .	3560
7.557	src/main/decaf/internal/util/concurrent/windows/MutexHandle.h File Reference .	3561
7.558	src/main/decaf/internal/util/HexStringParser.h File Reference	3561
7.559	src/main/decaf/internal/util/TimerTaskHeap.h File Reference	3561
7.560	src/main/decaf/io/BlockingByteArrayInputStream.h File Reference	3562
7.561	src/main/decaf/io/BufferedInputStream.h File Reference	3562
7.562	src/main/decaf/io/BufferedOutputStream.h File Reference	3563
7.563	src/main/decaf/io/ByteArrayInputStream.h File Reference	3563
7.564	src/main/decaf/io/ByteArrayOutputStream.h File Reference	3564
7.565	src/main/decaf/io/DataInputStream.h File Reference	3564
7.566	src/main/decaf/io/DataOutputStream.h File Reference	3565
7.567	src/main/decaf/io/EOFException.h File Reference	3565
7.568	src/main/decaf/io/FilterInputStream.h File Reference	3566
7.569	src/main/decaf/io/FilterOutputStream.h File Reference	3566
7.570	src/main/decaf/io/InputStream.h File Reference	3567
7.571	src/main/decaf/io/InterruptedIOException.h File Reference	3567
7.572	src/main/decaf/io/IOException.h File Reference	3568
7.573	src/main/decaf/io/OutputStream.h File Reference	3568
7.574	src/main/decaf/io/Reader.h File Reference	3569
7.575	src/main/decaf/io/UnsupportedEncodingException.h File Reference	3569
7.576	src/main/decaf/io/UTFDataFormatException.h File Reference	3569
7.577	src/main/decaf/io/Writer.h File Reference	3570
7.578	src/main/decaf/lang/Appendable.h File Reference	3570
7.579	src/main/decaf/lang/Boolean.h File Reference	3571
7.580	src/main/decaf/lang/Byte.h File Reference	3571
7.581	src/main/decaf/lang/Character.h File Reference	3572
7.582	src/main/decaf/lang/CharSequence.h File Reference	3572
7.583	src/main/decaf/lang/Comparable.h File Reference	3572
7.584	src/main/decaf/lang/Double.h File Reference	3573
7.585	src/main/decaf/lang/Exception.h File Reference	3573
7.586	src/main/decaf/lang/exceptions/ClassCastException.h File Reference	3574
7.587	src/main/decaf/lang/exceptions/IllegalArgumentException.h File Reference . . .	3574
7.588	src/main/decaf/lang/exceptions/IllegalMonitorStateException.h File Reference .	3575
7.589	src/main/decaf/lang/exceptions/IllegalThreadStateException.h File Reference .	3575
7.590	src/main/decaf/lang/exceptions/IndexOutOfBoundsException.h File Reference .	3575
7.591	src/main/decaf/lang/exceptions/InterruptedException.h File Reference	3576

7.592	src/main/decaf/lang/exceptions/InvalidStateException.h File Reference	3576
7.593	src/main/decaf/lang/exceptions/NoSuchElementException.h File Reference	3577
7.594	src/main/decaf/lang/exceptions/NullPointerException.h File Reference	3577
7.595	src/main/decaf/lang/exceptions/NumberFormatException.h File Reference	3577
7.596	src/main/decaf/lang/exceptions/RuntimeException.h File Reference	3578
7.597	src/main/decaf/lang/Float.h File Reference	3578
7.598	src/main/decaf/lang/Integer.h File Reference	3579
7.599	src/main/decaf/lang/Iterable.h File Reference	3579
7.600	src/main/decaf/lang/Long.h File Reference	3579
7.601	src/main/decaf/lang/Math.h File Reference	3580
7.602	src/main/decaf/lang/Number.h File Reference	3580
7.603	src/main/decaf/lang/Pointer.h File Reference	3581
7.604	src/main/decaf/lang/Runnable.h File Reference	3582
7.605	src/main/decaf/lang/Runtime.h File Reference	3582
7.606	src/main/decaf/lang/Short.h File Reference	3583
7.607	src/main/decaf/lang/System.h File Reference	3583
7.608	src/main/decaf/lang/Thread.h File Reference	3583
7.609	src/main/decaf/lang/ThreadGroup.h File Reference	3584
7.610	src/main/decaf/lang/Throwable.h File Reference	3584
7.611	src/main/decaf/net/BindException.h File Reference	3585
7.612	src/main/decaf/net/BufferedSocket.h File Reference	3585
7.613	src/main/decaf/net/ConnectException.h File Reference	3586
7.614	src/main/decaf/net/HttpRetryException.h File Reference	3586
7.615	src/main/decaf/net/MalformedURLException.h File Reference	3587
7.616	src/main/decaf/net/NoRouteToHostException.h File Reference	3587
7.617	src/main/decaf/net/PortUnreachableException.h File Reference	3587
7.618	src/main/decaf/net/ProtocolException.h File Reference	3588
7.619	src/main/decaf/net/ServerSocket.h File Reference	3588
7.620	src/main/decaf/net/Socket.h File Reference	3589
7.621	src/main/decaf/net/SocketError.h File Reference	3589
7.622	src/main/decaf/net/SocketException.h File Reference	3590
7.623	src/main/decaf/net/SocketFactory.h File Reference	3590
7.624	src/main/decaf/net/SocketInputStream.h File Reference	3591
7.625	src/main/decaf/net/SocketOutputStream.h File Reference	3591
7.626	src/main/decaf/net/SocketTimeoutException.h File Reference	3592
7.627	src/main/decaf/net/TcpSocket.h File Reference	3592

7.628src/main/decaf/net/UnknownHostException.h File Reference	3593
7.629src/main/decaf/net/UnknownServiceException.h File Reference	3593
7.630src/main/decaf/net/URI.h File Reference	3593
7.631src/main/decaf/net/URISyntaxException.h File Reference	3594
7.632src/main/decaf/net/URL.h File Reference	3594
7.633src/main/decaf/net/URLDecoder.h File Reference	3595
7.634src/main/decaf/net/URLEncoder.h File Reference	3595
7.635src/main/decaf/nio/Buffer.h File Reference	3596
7.636src/main/decaf/nio/BufferOverflowException.h File Reference	3596
7.637src/main/decaf/nio/BufferUnderflowException.h File Reference	3596
7.638src/main/decaf/nio/ByteBuffer.h File Reference	3597
7.639src/main/decaf/nio/CharBuffer.h File Reference	3597
7.640src/main/decaf/nio/DoubleBuffer.h File Reference	3598
7.641src/main/decaf/nio/FloatBuffer.h File Reference	3599
7.642src/main/decaf/nio/IntBuffer.h File Reference	3599
7.643src/main/decaf/nio/InvalidMarkException.h File Reference	3600
7.644src/main/decaf/nio/LongBuffer.h File Reference	3600
7.645src/main/decaf/nio/ReadOnlyBufferException.h File Reference	3601
7.646src/main/decaf/nio/ShortBuffer.h File Reference	3601
7.647src/main/decaf/security/auth/x500/X500Principal.h File Reference	3602
7.648src/main/decaf/security/cert/Certificate.h File Reference	3602
7.649src/main/decaf/security/cert/CertificateEncodingException.h File Reference . . .	3603
7.650src/main/decaf/security/cert/CertificateException.h File Reference	3603
7.651src/main/decaf/security/cert/CertificateExpiredException.h File Reference	3604
7.652src/main/decaf/security/cert/CertificateNotYetValidException.h File Reference . .	3604
7.653src/main/decaf/security/cert/CertificateParsingException.h File Reference	3604
7.654src/main/decaf/security/cert/X509Certificate.h File Reference	3605
7.655src/main/decaf/security/GeneralSecurityException.h File Reference	3605
7.656src/main/decaf/security/InvalidKeyException.h File Reference	3606
7.657src/main/decaf/security/Key.h File Reference	3606
7.658src/main/decaf/security/KeyException.h File Reference	3607
7.659src/main/decaf/security/NoSuchAlgorithmException.h File Reference	3607
7.660src/main/decaf/security/NoSuchProviderException.h File Reference	3607
7.661src/main/decaf/security/Principal.h File Reference	3608
7.662src/main/decaf/security/PublicKey.h File Reference	3608
7.663src/main/decaf/security/SignatureException.h File Reference	3609

7.664	src/main/decaf/security_provider/SecurityProvider.h File Reference	3609
7.665	src/main/decaf/security_provider/SecurityProviderMap.h File Reference	3609
7.666	src/main/decaf/security_provider/SecurityProviderRegistrar.h File Reference . . .	3610
7.667	src/main/decaf/security_provider/unix/openssl/OpenSSLX500Principal.h File Reference	3610
7.668	src/main/decaf/security_provider/unix/openssl/OpenSSLX509Certificate.h File Reference	3611
7.669	src/main/decaf/util/AbstractCollection.h File Reference	3611
7.670	src/main/decaf/util/AbstractList.h File Reference	3612
7.671	src/main/decaf/util/AbstractMap.h File Reference	3612
7.672	src/main/decaf/util/AbstractQueue.h File Reference	3613
7.673	src/main/decaf/util/AbstractSequentialList.h File Reference	3614
7.674	src/main/decaf/util/AbstractSet.h File Reference	3614
7.675	src/main/decaf/util/Collection.h File Reference	3615
7.676	src/main/decaf/util/Comparator.h File Reference	3615
7.677	src/main/decaf/util/comparators/Less.h File Reference	3616
7.678	src/main/decaf/util/concurrent/atomic/AtomicBoolean.h File Reference	3616
7.679	src/main/decaf/util/concurrent/atomic/AtomicInteger.h File Reference	3617
7.680	src/main/decaf/util/concurrent/atomic/AtomicReference.h File Reference	3617
7.681	src/main/decaf/util/concurrent/BlockingQueue.h File Reference	3618
7.682	src/main/decaf/util/concurrent/BrokenBarrierException.h File Reference	3618
7.683	src/main/decaf/util/concurrent/Callable.h File Reference	3619
7.684	src/main/decaf/util/concurrent/CancellationException.h File Reference	3619
7.685	src/main/decaf/util/concurrent/Concurrent.h File Reference	3619
7.685.1	Define Documentation	3620
7.685.1.1	synchronized	3620
7.685.1.2	WAIT_INFINITE	3620
7.686	src/main/decaf/util/concurrent/ConcurrentMap.h File Reference	3620
7.687	src/main/decaf/util/concurrent/ConcurrentStlMap.h File Reference	3621
7.688	src/main/decaf/util/concurrent/CountDownLatch.h File Reference	3622
7.689	src/main/decaf/util/concurrent/Delayed.h File Reference	3622
7.690	src/main/decaf/util/concurrent/ExecutionException.h File Reference	3623
7.691	src/main/decaf/util/concurrent/Executor.h File Reference	3623
7.692	src/main/decaf/util/concurrent/ExecutorService.h File Reference	3623
7.693	src/main/decaf/util/concurrent/Future.h File Reference	3624
7.694	src/main/decaf/util/concurrent/Lock.h File Reference	3624
7.695	src/main/decaf/util/concurrent/locks/Lock.h File Reference	3625

7.696	src/main/decaf/util/concurrent/locks/Condition.h File Reference	3625
7.697	src/main/decaf/util/concurrent/locks/LockSupport.h File Reference	3626
7.698	src/main/decaf/util/concurrent/locks/ReadWriteLock.h File Reference	3627
7.699	src/main/decaf/util/concurrent/locks/ReentrantLock.h File Reference	3627
7.700	src/main/decaf/util/concurrent/Mutex.h File Reference	3628
7.701	src/main/decaf/util/concurrent/PooledThread.h File Reference	3628
7.702	src/main/decaf/util/concurrent/PooledThreadListener.h File Reference	3629
7.703	src/main/decaf/util/concurrent/RejectedExecutionException.h File Reference . . .	3629
7.704	src/main/decaf/util/concurrent/RejectedExecutionHandler.h File Reference . . .	3629
7.705	src/main/decaf/util/concurrent/Semaphore.h File Reference	3630
7.706	src/main/decaf/util/concurrent/Synchronizable.h File Reference	3630
7.707	src/main/decaf/util/concurrent/SynchronousQueue.h File Reference	3631
7.708	src/main/decaf/util/concurrent/TaskListener.h File Reference	3632
7.709	src/main/decaf/util/concurrent/ThreadFactory.h File Reference	3632
7.710	src/main/decaf/util/concurrent/ThreadPool.h File Reference	3632
7.711	src/main/decaf/util/concurrent/TimeoutException.h File Reference	3633
7.712	src/main/decaf/util/concurrent/TimeUnit.h File Reference	3634
7.713	src/main/decaf/util/Date.h File Reference	3634
7.714	src/main/decaf/util/Iterator.h File Reference	3635
7.715	src/main/decaf/util/List.h File Reference	3635
7.716	src/main/decaf/util/ListIterator.h File Reference	3636
7.717	src/main/decaf/util/logging/ConsoleHandler.h File Reference	3636
7.718	src/main/decaf/util/logging/Filter.h File Reference	3636
7.719	src/main/decaf/util/logging/Formatter.h File Reference	3637
7.720	src/main/decaf/util/logging/Handler.h File Reference	3637
7.721	src/main/decaf/util/logging/Logger.h File Reference	3638
7.722	src/main/decaf/util/logging/LoggerCommon.h File Reference	3638
7.723	src/main/decaf/util/logging/LoggerDefines.h File Reference	3639
7.723.1	Define Documentation	3639
7.723.1.1	LOGDECAF_DEBUG	3639
7.723.1.2	LOGDECAF_DEBUG_1	3639
7.723.1.3	LOGDECAF_DECLARE	3640
7.723.1.4	LOGDECAF_DECLARE_LOCAL	3640
7.723.1.5	LOGDECAF_ERROR	3640
7.723.1.6	LOGDECAF_FATAL	3640
7.723.1.7	LOGDECAF_INFO	3640

7.723.1.8 LOGDECAF_ INITIALIZE	3640
7.723.1.9 LOGDECAF_ WARN	3640
7.724src/main/decaf/util/logging/LoggerHierarchy.h File Reference	3640
7.725src/main/decaf/util/logging/LogManager.h File Reference	3640
7.726src/main/decaf/util/logging/LogRecord.h File Reference	3641
7.727src/main/decaf/util/logging/LogWriter.h File Reference	3641
7.728src/main/decaf/util/logging/MarkBlockLogger.h File Reference	3642
7.729src/main/decaf/util/logging/PropertiesChangeListener.h File Reference	3642
7.730src/main/decaf/util/logging/SimpleFormatter.h File Reference	3643
7.731src/main/decaf/util/logging/SimpleLogger.h File Reference	3643
7.732src/main/decaf/util/logging/StreamHandler.h File Reference	3644
7.733src/main/decaf/util/Map.h File Reference	3644
7.734src/main/decaf/util/PriorityQueue.h File Reference	3645
7.735src/main/decaf/util/Properties.h File Reference	3645
7.736src/main/decaf/util/Random.h File Reference	3646
7.737src/main/decaf/util/Set.h File Reference	3647
7.738src/main/decaf/util/StlList.h File Reference	3647
7.739src/main/decaf/util/StlMap.h File Reference	3648
7.740src/main/decaf/util/StlQueue.h File Reference	3648
7.741src/main/decaf/util/StlSet.h File Reference	3649
7.742src/main/decaf/util/StringTokenizer.h File Reference	3650
7.743src/main/decaf/util/Timer.h File Reference	3650
7.744src/main/decaf/util/TimerTask.h File Reference	3651
7.745src/main/decaf/util/UUID.h File Reference	3651

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

activemq (Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements)	69
activemq::cmsutil	70
activemq::commands	71
activemq::core	72
activemq::exceptions	73
activemq::io	73
activemq::library	73
activemq::state	73
activemq::threads	74
activemq::transport	74
activemq::transport::correlator	75
activemq::transport::failover	75
activemq::transport::inactivity	76
activemq::transport::logging	76
activemq::transport::mock	76
activemq::transport::tcp	77
activemq::util	77
activemq::wireformat	78
activemq::wireformat::openwire	78
activemq::wireformat::openwire::marshal	78
activemq::wireformat::openwire::marshal::v1	79
activemq::wireformat::openwire::marshal::v2	83
activemq::wireformat::openwire::marshal::v3	87
activemq::wireformat::openwire::marshal::v4	90
activemq::wireformat::openwire::marshal::v5	94
activemq::wireformat::openwire::utils	98
activemq::wireformat::stomp	99
cms (Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements)	99
decaf (Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements)	102
decaf::internal	102

<code>decaf::internal::io</code>	103
<code>decaf::internal::net</code>	103
<code>decaf::internal::nio</code>	103
<code>decaf::internal::util</code>	104
<code>decaf::internal::util::concurrent</code>	104
<code>decaf::io</code>	105
<code>decaf::lang</code>	106
<code>decaf::lang::exceptions</code>	108
<code>decaf::net</code>	108
<code>decaf::nio</code>	109
<code>decaf::security</code>	110
<code>decaf::security::auth</code>	110
<code>decaf::security::auth::x500</code>	110
<code>decaf::security::cert</code>	110
<code>decaf::security_provider</code>	111
<code>decaf::security_provider::unix</code>	111
<code>decaf::security_provider::unix::openssl</code>	111
<code>decaf::util</code>	112
<code>decaf::util::comparators</code>	114
<code>decaf::util::concurrent</code>	114
<code>decaf::util::concurrent::atomic</code>	115
<code>decaf::util::concurrent::locks</code>	116
<code>decaf::util::logging</code>	116
<code>std</code>	117

Chapter 2

Data Structure Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

activemq::core::ActiveMQAckHandler	141
activemq::core::ActiveMQConstants	227
activemq::library::ActiveMQCPP	238
activemq::core::ActiveMQTransactionContext	573
decaf::lang::Appendable	576
decaf::nio::CharBuffer	930
decaf::internal::nio::CharArrayBuffer	922
decaf::internal::AprPool	578
decaf::util::concurrent::atomic::AtomicBoolean	579
decaf::lang::AtomicRefCounter	587
decaf::util::concurrent::atomic::AtomicReference< T >	589
binary_function	663
decaf::util::Comparator< E >	1011
decaf::util::comparators::Less< E >	1862
decaf::util::Comparator< Pointer< T, R > >	1011
decaf::lang::PointerComparator< T, R >	2350
std::less< decaf::lang::Pointer< T > >	1864
activemq::wireformat::openwire::utils::BooleanStream	680
decaf::nio::Buffer	741
decaf::nio::ByteBuffer	848
decaf::internal::nio::ByteArrayBuffer	806
decaf::nio::CharBuffer	930
decaf::nio::DoubleBuffer	1459
decaf::internal::nio::DoubleArrayBuffer	1452
decaf::nio::FloatBuffer	1562
decaf::internal::nio::FloatArrayBuffer	1555
decaf::nio::IntBuffer	1641
decaf::internal::nio::IntArrayBuffer	1634
decaf::nio::LongBuffer	1955
decaf::internal::nio::LongArrayBuffer	1948
decaf::nio::ShortBuffer	2745
decaf::internal::nio::ShortArrayBuffer	2738

decaf::internal::nio::BufferFactory	762
decaf::internal::util::ByteArrayAdapter	784
decaf::internal::nio::ByteArrayPerspective	843
decaf::util::concurrent::Callable< V >	897
decaf::security::cert::Certificate	901
decaf::security::cert::X509Certificate	3189
decaf::security_provider::unix::openssl::OpenSSLX509Certificate	2290
decaf::lang::CharSequence	946
decaf::nio::CharBuffer	930
decaf::io::Closeable	950
activemq::transport::Transport	3066
activemq::transport::CompositeTransport	1019
activemq::transport::failover::FailoverTransport	1512
activemq::transport::IOTransport	1709
activemq::transport::mock::MockTransport	2227
activemq::transport::TransportFilter	3073
activemq::transport::correlator::ResponseCorrelator	2616
activemq::transport::inactivity::InactivityMonitor	1624
activemq::transport::logging::LoggingTransport	1920
activemq::transport::tcp::TcpTransport	2967
activemq::wireformat::WireFormatNegotiator	3182
activemq::wireformat::openwire::OpenWireFormatNegotiator	2310
decaf::io::InputStream	1630
decaf::internal::io::StandardInputStream	2818
decaf::io::BlockingByteArrayInputStream	666
decaf::io::ByteArrayInputStream	828
decaf::io::FilterInputStream	1528
activemq::io::LoggingInputStream	1917
decaf::io::BufferedInputStream	747
decaf::io::DataInputStream	1291
decaf::net::SocketInputStream	2796
decaf::io::OutputStream	2316
decaf::internal::io::StandardErrorOutputStream	2812
decaf::internal::io::StandardOutputStream	2825
decaf::io::ByteArrayOutputStream	836
decaf::io::FilterOutputStream	1536
activemq::io::LoggingOutputStream	1919
decaf::io::BufferedOutputStream	752
decaf::io::DataOutputStream	1300
decaf::net::SocketOutputStream	2803
decaf::net::Socket	2786
decaf::net::BufferedSocket	755
decaf::net::TcpSocket	2959
decaf::util::logging::Handler	1604
decaf::util::logging::StreamHandler	2888
cms::Closeable	951
activemq::commands::ActiveMQTempDestination	455
activemq::commands::ActiveMQTempQueue	477
activemq::commands::ActiveMQTempTopic	501
cms::Connection	1052
activemq::core::ActiveMQConnection	202

cms::MessageConsumer	2080
activemq::cmsutil::CachedConsumer	888
activemq::core::ActiveMQConsumer	230
cms::MessageProducer	2189
activemq::cmsutil::CachedProducer	891
activemq::core::ActiveMQProducer	367
cms::QueueBrowser	2511
cms::Session	2663
activemq::cmsutil::PooledSession	2351
activemq::core::ActiveMQSession	402
activemq::cmsutil::CmsAccessor	954
activemq::cmsutil::CmsDestinationAccessor	957
activemq::cmsutil::CmsTemplate	969
cms::CMSException	960
cms::CMSSecurityException	968
cms::IllegalStateException	1621
cms::InvalidClientIdException	1696
cms::InvalidDestinationException	1697
cms::InvalidSelectorException	1703
cms::MessageEOFException	2140
cms::MessageFormatException	2141
cms::MessageNotReadableException	2187
cms::MessageNotWriteableException	2188
cms::UnsupportedOperationException	3093
activemq::util::CMSExceptionSupport	963
cms::CMSProperties	965
activemq::util::ActiveMQProperties	374
activemq::state::CommandVisitor	996
activemq::state::CommandVisitorAdapter	1003
activemq::state::ConnectionStateTracker	1160
decaf::lang::Comparable< T >	1009
decaf::lang::Comparable< bool >	1009
decaf::lang::Boolean	674
decaf::lang::Comparable< Boolean >	1009
decaf::lang::Boolean	674
decaf::lang::Comparable< BrokerId >	1009
activemq::commands::BrokerId	691
decaf::lang::Comparable< Byte >	1009
decaf::lang::Byte	776
decaf::lang::Comparable< ByteBuffer >	1009
decaf::nio::ByteBuffer	848
decaf::lang::Comparable< char >	1009
decaf::lang::Character	914
decaf::lang::Comparable< Character >	1009
decaf::lang::Character	914
decaf::lang::Comparable< CharBuffer >	1009
decaf::nio::CharBuffer	930
decaf::lang::Comparable< ConnectionId >	1009
activemq::commands::ConnectionId	1106

decaf::lang::Comparable< ConsumerId >	1009
activemq::commands::ConsumerId	1190
decaf::lang::Comparable< Date >	1009
decaf::util::Date	1377
decaf::lang::Comparable< Delayed >	1009
decaf::util::concurrent::Delayed	1385
decaf::lang::Comparable< Double >	1009
decaf::lang::Double	1441
decaf::lang::Comparable< double >	1009
decaf::lang::Double	1441
decaf::lang::Comparable< DoubleBuffer >	1009
decaf::nio::DoubleBuffer	1459
decaf::lang::Comparable< float >	1009
decaf::lang::Float	1544
decaf::lang::Comparable< Float >	1009
decaf::lang::Float	1544
decaf::lang::Comparable< FloatBuffer >	1009
decaf::nio::FloatBuffer	1562
decaf::lang::Comparable< int >	1009
decaf::lang::Integer	1652
decaf::lang::Comparable< IntBuffer >	1009
decaf::nio::IntBuffer	1641
decaf::lang::Comparable< Integer >	1009
decaf::lang::Integer	1652
decaf::lang::Comparable< LocalTransactionId >	1009
activemq::commands::LocalTransactionId	1874
decaf::lang::Comparable< Long >	1009
decaf::lang::Long	1934
decaf::lang::Comparable< long long >	1009
decaf::lang::Long	1934
decaf::lang::Comparable< LongBuffer >	1009
decaf::nio::LongBuffer	1955
decaf::lang::Comparable< MessageId >	1009
activemq::commands::MessageId	2142
decaf::lang::Comparable< ProducerId >	1009
activemq::commands::ProducerId	2446
decaf::lang::Comparable< SessionId >	1009
activemq::commands::SessionId	2677
decaf::lang::Comparable< Short >	1009
decaf::lang::Short	2729
decaf::lang::Comparable< short >	1009
decaf::lang::Short	2729
decaf::lang::Comparable< ShortBuffer >	1009
decaf::nio::ShortBuffer	2745
decaf::lang::Comparable< TimeUnit >	1009
decaf::util::concurrent::TimeUnit	3005

decaf::lang::Comparable< TransactionId >	1009
activemq::commands::TransactionId	3015
activemq::commands::LocalTransactionId	1874
activemq::commands::XATransactionId	3192
decaf::lang::Comparable< unsigned char >	1009
decaf::lang::Byte	776
decaf::lang::Comparable< URI >	1009
decaf::net::URI	3096
decaf::lang::Comparable< UUID >	1009
decaf::util::UUID	3141
decaf::lang::Comparable< XATransactionId >	1009
activemq::commands::XATransactionId	3192
decaf::util::Comparator< T >	1011
activemq::util::CompositeData	1013
decaf::util::concurrent::locks::Condition	1040
decaf::util::concurrent::ConditionHandle	1046
decaf::internal::util::concurrent::ConditionImpl	1047
cms::ConnectionFactory	1103
activemq::core::ActiveMQConnectionFactory	211
cms::ConnectionMetaData	1154
activemq::core::ActiveMQConnectionMetaData	216
activemq::state::ConnectionState	1158
activemq::state::ConsumerState	1242
decaf::util::concurrent::CountDownLatch	1266
activemq::wireformat::openwire::marshal::DataStreamMarshaller	1330
activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller	636
activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller	254
activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller	386
activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller	462
activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller	485
activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller	509
activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller	557
activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller	610
activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller	725
activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller	1063
activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller	1087
activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller	1142
activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller	1178
activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller	1226
activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller	1254
activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller	1407
activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller	1579
activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller	1831
activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller	2076
activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller	2107
activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller	2132
activemq::wireformat::openwire::marshal::v1::MessageMarshaller	2183
activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller	150
activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller	186
activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller	288
activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller	311

activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller	351
activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller	439
activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller	533
activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller	2215
activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller	2439
activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller	2490
activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller	2540
activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller	2572
activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller	2603
activemq::wireformat::openwire::marshal::v1::ResponseMarshaller	2633
activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller	1279
activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller	1314
activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller	1491
activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller	1673
activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller	2709
activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller	2760
activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller	3048
activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller	698
activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller	1113
activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller	1202
activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller	1435
activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller	1733
activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller	1750
activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller	1781
activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller	1804
activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller	2161
activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller	2261
activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller	2335
activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller	1846
activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller	2461
activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller	2685
activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller	2911
activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller	3022
activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller	1890
activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller	3209
activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller	3170
activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller	265
activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	398
activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller	473
activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller	497
activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller	521
activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller	569
activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller	630
activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller	737
activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller	1075
activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller	1099
activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller	1150
activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller	1186
activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller	1238
activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller	1262
activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller	1395
activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller	1576
activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller	1815

activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller	2060
activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller	2095
activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller	2120
activemq::wireformat::openwire::marshal::v2::MessageMarshaller	2166
activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller	162
activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller	198
activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller	300
activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller	323
activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller	363
activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller	451
activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller	545
activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller	2207
activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller	2424
activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller	2474
activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller	2548
activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller	2568
activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller	2587
activemq::wireformat::openwire::marshal::v2::ResponseMarshaller	2620
activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller	1287
activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller	1326
activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller	1487
activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller	1669
activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller	2705
activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller	2771
activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller	3056
activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller	709
activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller	1124
activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller	1210
activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller	1420
activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller	1722
activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller	1746
activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller	1769
activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller	1793
activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller	2146
activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller	2250
activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller	2322
activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller	1842
activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller	2450
activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller	2689
activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller	2926
activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller	3018
activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller	1878
activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller	3197
activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller	3163
activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller	250
activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller	382
activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller	459
activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller	481
activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller	505
activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller	553
activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller	603
activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller	721
activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller	1059

activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller	1083
activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller	1134
activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller	1170
activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller	1223
activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller	1246
activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller	1399
activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller	1583
activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller	1823
activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller	2068
activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller	2103
activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller	2128
activemq::wireformat::openwire::marshal::v3::MessageMarshaller	2175
activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller	146
activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller	182
activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller	284
activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller . .	307
activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller	347
activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller	436
activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller	529
activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller	2211
activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller	2427
activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller	2478
activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller	2552
activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller	2576
activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller	2591
activemq::wireformat::openwire::marshal::v3::ResponseMarshaller	2624
activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller	1271
activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller	1310
activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller	1495
activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller . .	1677
activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller	2721
activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller	2767
activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller	3044
activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller	694
activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller	1109
activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller	1195
activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller	1424
activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller	1729
activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller	1754
activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller	1777
activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller	1797
activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller	2154
activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller	2253
activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller	2326
activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller .	1850
activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller	2454
activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller	2697
activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller	2915
activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller	3033
activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller . .	1882
activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller . . .	3201
activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller	3178
activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller	258

activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller	390
activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller	466
activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller	489
activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller	513
activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller	561
activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller	616
activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller	729
activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller . .	1067
activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller . . .	1091
activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller	1138
activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller . . .	1174
activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller	1230
activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller . . .	1250
activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller	1403
activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller	1587
activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller	1827
activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller	2072
activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller . . .	2099
activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller	2124
activemq::wireformat::openwire::marshal::v4::MessageMarshaller	2170
activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller	154
activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller	190
activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller	292
activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller .	315
activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller	355
activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller	443
activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller	537
activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller	2223
activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller	2431
activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller	2486
activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller	2556
activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller	2580
activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller . . .	2599
activemq::wireformat::openwire::marshal::v4::ResponseMarshaller	2637
activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller	1275
activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller . . .	1322
activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller	1499
activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller .	1681
activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller	2713
activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller	2763
activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller	3052
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller	701
activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller	1116
activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller	1198
activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller	1427
activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller	1725
activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller	1758
activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller	1773
activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller	1801
activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller	2150
activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller . . .	2265
activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller	2330
activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller	1854

activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller	2465
activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller	2693
activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller	2923
activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller	3026
activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller . . .	1886
activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller . . .	3205
activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller	3174
activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller . . .	262
activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller	394
activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller	470
activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller	493
activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller	517
activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller	565
activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller	623
activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller	733
activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller . .	1071
activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller . . .	1095
activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller	1146
activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller . . .	1182
activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller	1234
activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller . . .	1258
activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller	1411
activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller	1591
activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller	1819
activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller	2064
activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller . . .	2111
activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller	2136
activemq::wireformat::openwire::marshal::v5::MessageMarshaller	2179
activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller	158
activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller	194
activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller	296
activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller .	319
activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller	359
activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller	447
activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller	541
activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller	2219
activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller	2435
activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller	2482
activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller	2544
activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller	2564
activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller . . .	2595
activemq::wireformat::openwire::marshal::v5::ResponseMarshaller	2628
activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller	1283
activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller . . .	1318
activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller	1503
activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller . .	1685
activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller	2717
activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller	2775
activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller	3041
activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller	705
activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller	1120
activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller	1206
activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller	1431

activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller	1737
activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller	1762
activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller	1785
activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller	1808
activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller	2157
activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller	2257
activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller	2339
activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller	1858
activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller	2457
activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller	2681
activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller	2919
activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller	3029
activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller	1894
activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller	3213
activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller	3166
cms::DeliveryMode	1386
cms::Destination	1387
cms::Queue	2510
activemq::commands::ActiveMQQueue	379
cms::TemporaryQueue	2971
activemq::commands::ActiveMQTempQueue	477
cms::TemporaryTopic	2973
activemq::commands::ActiveMQTempTopic	501
cms::Topic	3014
activemq::commands::ActiveMQTopic	549
activemq::commands::ActiveMQDestination::DestinationFilter	1390
activemq::cmsutil::DestinationResolver	1415
activemq::cmsutil::DynamicDestinationResolver	1471
activemq::core::DispatchData	1439
activemq::core::Dispatcher	1440
activemq::core::ActiveMQConsumer	230
activemq::core::ActiveMQSession	402
decaf::lang::DYNAMIC_CAST_TOKEN	1471
decaf::util::Map< K, V, COMPARATOR >::Entry	1473
cms::ExceptionListener	1483
decaf::util::concurrent::Executor	1509
decaf::util::concurrent::ExecutorService	1511
decaf::util::logging::Filter	1527
decaf::util::logging::Formatter	1595
decaf::util::logging::SimpleFormatter	2782
decaf::util::concurrent::Future< V >	1597
activemq::transport::correlator::FutureResponse	1600
decaf::security::GeneralSecurityException	1602
decaf::security::cert::CertificateException	906
decaf::security::cert::CertificateEncodingException	904
decaf::security::cert::CertificateExpiredException	908
decaf::security::cert::CertificateNotYetValidException	910
decaf::security::cert::CertificateParsingException	912
decaf::security::KeyException	1837
decaf::security::InvalidKeyException	1698
decaf::security::NoSuchAlgorithmExceptionException	2272

decaf::security::NoSuchProviderException	2277
decaf::security::SignatureException	2779
decaf::internal::util::HexStringParser	1607
activemq::wireformat::openwire::utils::HexTable	1609
decaf::lang::Iterable< E >	1715
decaf::util::Collection< E >	982
decaf::util::AbstractCollection< E >	119
decaf::util::List< E >	1865
decaf::util::AbstractList< E >	131
decaf::util::AbstractSequentialList< E >	137
decaf::util::StlList< E >	2833
decaf::util::Queue< E >	2507
decaf::util::AbstractQueue< E >	133
decaf::util::PriorityQueue< E >	2413
decaf::util::Set< E >	2729
decaf::util::AbstractSet< E >	138
decaf::util::StlSet< E >	2865
decaf::lang::Iterable< PrimitiveValueNode >	1715
decaf::util::Collection< PrimitiveValueNode >	982
decaf::util::AbstractCollection< PrimitiveValueNode >	119
decaf::util::List< PrimitiveValueNode >	1865
decaf::util::StlList< PrimitiveValueNode >	2833
activemq::util::PrimitiveList	2370
decaf::util::Iterator< T >	1716
decaf::util::Iterator< E >	1716
decaf::util::ListIterator< E >	1871
decaf::security::Key	1835
decaf::security::PublicKey	2506
decaf::util::concurrent::locks::Lock	1898
decaf::util::concurrent::locks::ReentrantLock	2526
decaf::util::concurrent::Lock	1903
decaf::util::concurrent::locks::LockSupport	1905
decaf::util::logging::Logger	1908
decaf::util::logging::LoggerHierarchy	1916
decaf::util::logging::LogManager	1923
decaf::util::logging::LogRecord	1928
decaf::util::logging::LogWriter	1932
activemq::util::LongSequenceGenerator	1967
decaf::util::logging::MarkBlockLogger	1992
activemq::wireformat::MarshalAware	1992
activemq::commands::DataStructure	1372
activemq::commands::BaseDataStructure	659
activemq::commands::ActiveMQDestination	240
activemq::commands::ActiveMQQueue	379
activemq::commands::ActiveMQTempDestination	455
activemq::commands::ActiveMQTopic	549
activemq::commands::BooleanExpression	679
activemq::commands::BrokerId	691
activemq::commands::Command	991
activemq::commands::BaseCommand	596

activemq::commands::BrokerError	686
activemq::commands::BrokerInfo	713
activemq::commands::ConnectionControl	1055
activemq::commands::ConnectionError	1079
activemq::commands::ConnectionInfo	1128
activemq::commands::ConsumerControl	1165
activemq::commands::ConsumerInfo	1214
activemq::commands::ControlCommand	1243
activemq::commands::DestinationInfo	1390
activemq::commands::FlushCommand	1573
activemq::commands::KeepAliveInfo	1812
activemq::commands::Message	2018
activemq::commands::ActiveMQMessageTemplate< T >	327
activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >	327
activemq::commands::ActiveMQBytesMessage	166
activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >	327
activemq::commands::ActiveMQMapMessage	272
activemq::commands::ActiveMQMessageTemplate< cms::Message >	327
activemq::commands::ActiveMQBlobMessage	142
activemq::commands::ActiveMQMessage	304
activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >	327
activemq::commands::ActiveMQObjectMessage	344
activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >	327
activemq::commands::ActiveMQStreamMessage	422
activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >	327
activemq::commands::ActiveMQTextMessage	525
activemq::commands::MessageAck	2055
activemq::commands::MessageDispatch	2084
activemq::commands::MessageDispatchNotification	2115
activemq::commands::MessagePull	2202
activemq::commands::ProducerAck	2420
activemq::commands::ProducerInfo	2469
activemq::commands::RemoveInfo	2536
activemq::commands::RemoveSubscriptionInfo	2560
activemq::commands::ReplayCommand	2584
activemq::commands::Response	2611
activemq::commands::DataArrayResponse	1267
activemq::commands::DataResponse	1307
activemq::commands::ExceptionResponse	1484
activemq::commands::IntegerResponse	1667
activemq::state::Tracked	3015
activemq::commands::SessionInfo	2701
activemq::commands::ShutdownInfo	2757
activemq::commands::TransactionInfo	3037
activemq::commands::WireFormatInfo	3152
activemq::commands::ConnectionId	1106
activemq::commands::ConsumerId	1190
activemq::commands::DiscoveryEvent	1416
activemq::commands::JournalQueueAck	1718
activemq::commands::JournalTopicAck	1741
activemq::commands::JournalTrace	1766
activemq::commands::JournalTransaction	1789

activemq::commands::MessageId	2142
activemq::commands::NetworkBridgeFilter	2246
activemq::commands::PartialCommand	2318
activemq::commands::LastPartialCommand	1840
activemq::commands::ProducerId	2446
activemq::commands::SessionId	2677
activemq::commands::SubscriptionInfo	2907
activemq::commands::TransactionId	3015
activemq::wireformat::openwire::marshal::v2::MarshallerFactory	1995
activemq::wireformat::openwire::marshal::v4::MarshallerFactory	1996
activemq::wireformat::openwire::marshal::v1::MarshallerFactory	1997
activemq::wireformat::openwire::marshal::v3::MarshallerFactory	1997
activemq::wireformat::openwire::marshal::v5::MarshallerFactory	1998
decaf::lang::Math	1999
cms::Message	2036
activemq::commands::ActiveMQMessageTemplate< cms::Message >	327
cms::BytesMessage	874
activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >	327
cms::MapMessage	1982
activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >	327
cms::ObjectMessage	2287
activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >	327
cms::StreamMessage	2892
activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >	327
cms::TextMessage	2974
activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >	327
activemq::cmsutil::MessageCreator	2083
cms::MessageListener	2165
activemq::wireformat::openwire::utils::MessagePropertyInterceptor	2196
decaf::util::concurrent::MutexHandle	2244
decaf::internal::util::concurrent::MutexImpl	2244
decaf::lang::Number	2282
decaf::lang::Byte	776
decaf::lang::Character	914
decaf::lang::Double	1441
decaf::lang::Float	1544
decaf::lang::Integer	1652
decaf::lang::Long	1934
decaf::lang::Short	2729
decaf::util::concurrent::atomic::AtomicInteger	582
decaf::security_provider::unix::openssl::OpenSSLX500Principal	2287
activemq::wireformat::openwire::utils::OpenwireStringSupport	2315
decaf::lang::Pointer< T, REFCOUNTER >	2343
decaf::util::concurrent::PooledThread	2364
decaf::util::concurrent::PooledThreadListener	2366
decaf::util::concurrent::ThreadPool	2977
activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller	2390
activemq::util::PrimitiveValueNode::PrimitiveValue	2395
activemq::util::PrimitiveValueConverter	2397
activemq::util::PrimitiveValueNode	2398
decaf::security::Principal	2411
decaf::security::auth::x500::X500Principal	3188

activemq::cmsutil::ProducerCallback	2443
activemq::cmsutil::CmsTemplate::SendExecutor	2660
activemq::state::ProducerState	2494
decaf::util::Properties	2494
decaf::util::logging::PropertiesChangeListener	2503
decaf::util::Random	2513
decaf::io::Reader	2518
decaf::util::concurrent::locks::ReadWriteLock	2522
decaf::util::concurrent::RejectedExecutionHandler	2536
activemq::cmsutil::ResourceLifecycleManager	2608
activemq::transport::mock::ResponseBuilder	2614
activemq::wireformat::openwire::OpenWireResponseBuilder	2313
decaf::lang::Runnable	2642
activemq::threads::CompositeTaskRunner	1017
activemq::threads::DedicatedTaskRunner	1382
activemq::transport::IOTransport	1709
decaf::util::TimerTask	3000
activemq::transport::inactivity::ReadChecker	2517
activemq::transport::inactivity::WriteChecker	3186
decaf::lang::Runtime	2643
decaf::internal::DecafRuntime	1381
decaf::security_provider::SecurityProvider	2647
decaf::security_provider::SecurityProviderMap	2648
decaf::security_provider::SecurityProviderRegistrar	2649
decaf::util::concurrent::Semaphore	2651
decaf::net::ServerSocket	2661
activemq::cmsutil::SessionCallback	2676
activemq::cmsutil::CmsTemplate::ProducerExecutor	2444
activemq::cmsutil::CmsTemplate::ResolveProducerExecutor	2607
activemq::cmsutil::CmsTemplate::ReceiveExecutor	2524
activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor	2608
activemq::cmsutil::SessionPool	2725
activemq::state::SessionState	2726
decaf::util::logging::SimpleLogger	2784
decaf::net::SocketError	2792
decaf::net::SocketFactory	2795
activemq::commands::BrokerError::StackTraceElement	2811
cms::Startable	2831
cms::Connection	1052
decaf::lang::STATIC_CAST_TOKEN	2831
activemq::core::ActiveMQConstants::StaticInitializer	2832
activemq::wireformat::stomp::StompCommandConstants	2871
activemq::wireformat::stomp::StompFrame	2874
activemq::wireformat::stomp::StompHelper	2878
cms::Stoppable	2887
cms::Connection	1052
decaf::util::StringTokenizer	2904
decaf::util::concurrent::Synchronizable	2930
activemq::core::MessageDispatchChannel	2088
decaf::util::Collection< PrimitiveValueNode >	982
decaf::internal::util::concurrent::SynchronizableImpl	2941

decaf::io::InputStream	1630
decaf::io::OutputStream	2316
decaf::util::Collection< E >	982
decaf::util::concurrent::Mutex	2239
decaf::util::Map< K, V, COMPARATOR >	1970
decaf::util::AbstractMap< K, V, COMPARATOR >	132
decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR >	1020
decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >	1024
decaf::util::StlMap< K, V, COMPARATOR >	2845
decaf::util::StlQueue< T >	2858
decaf::util::Map< std::string, PrimitiveValueNode, std::less< std::string > >	1970
decaf::util::StlMap< std::string, PrimitiveValueNode >	2845
activemq::util::PrimitiveMap	2381
activemq::core::Synchronization	2945
decaf::util::concurrent::SynchronousQueue< E >	2946
decaf::lang::System	2953
activemq::threads::Task	2956
activemq::core::ActiveMQSessionExecutor	418
activemq::threads::CompositeTask	1016
activemq::transport::failover::BackupTransportPool	594
activemq::transport::failover::CloseTransportsTask	952
activemq::transport::failover::FailoverTransport	1512
activemq::threads::CompositeTaskRunner	1017
decaf::util::concurrent::TaskListener	2957
activemq::threads::TaskRunner	2958
activemq::threads::CompositeTaskRunner	1017
activemq::threads::DedicatedTaskRunner	1382
decaf::util::concurrent::ThreadFactory	2976
decaf::lang::ThreadGroup	2977
decaf::lang::Throwable	2983
decaf::lang::Exception	1476
activemq::exceptions::ActiveMQException	269
activemq::exceptions::BrokerException	690
decaf::io::IOException	1707
decaf::io::EOFException	1474
decaf::io::InterruptedIOException	1693
decaf::net::SocketTimeoutException	2809
decaf::io::UnsupportedEncodingException	3090
decaf::io::UTFDataFormatException	3139
decaf::net::HttpRetryException	1610
decaf::net::MalformedURLException	1968
decaf::net::ProtocolException	2504
decaf::net::SocketException	2793
decaf::net::BindException	663
decaf::net::ConnectException	1049
decaf::net::NoRouteToHostException	2269
decaf::net::PortUnreachableException	2368
decaf::net::UnknownHostException	3085
decaf::net::UnknownServiceException	3087
decaf::lang::exceptions::ClassCastException	948
decaf::lang::exceptions::IllegalArgumentException	1613
decaf::lang::exceptions::IllegalMonitorStateException	1616

decaf::lang::exceptions::IllegalStateException	1618
decaf::nio::InvalidMarkException	1700
decaf::lang::exceptions::IllegalThreadStateException	1622
decaf::lang::exceptions::IndexOutOfBoundsException	1627
decaf::lang::exceptions::InterruptedException	1691
decaf::lang::exceptions::InvalidStateException	1704
decaf::lang::exceptions::NoSuchElementException	2274
decaf::lang::exceptions::NullPointerException	2279
decaf::lang::exceptions::NumberFormatException	2284
decaf::lang::exceptions::RuntimeException	2644
decaf::lang::exceptions::UnsupportedOperationException	3094
decaf::nio::ReadOnlyBufferException	2520
decaf::net::URISyntaxException	3122
decaf::nio::BufferOverflowException	771
decaf::nio::BufferUnderflowException	774
decaf::util::concurrent::BrokenBarrierException	683
decaf::util::concurrent::CancellationException	898
decaf::util::concurrent::ExecutionException	1507
decaf::util::concurrent::RejectedExecutionException	2533
decaf::util::concurrent::TimeoutException	2987
decaf::util::Timer	2989
decaf::internal::util::TimerTaskHeap	3002
activemq::state::TransactionState	3060
decaf::internal::util::concurrent::Transferer< E >	3062
decaf::internal::util::concurrent::TransferQueue< E >	3062
decaf::internal::util::concurrent::TransferStack< E >	3064
activemq::transport::TransportFactory	3071
activemq::transport::AbstractTransportFactory	139
activemq::transport::failover::FailoverTransportFactory	1523
activemq::transport::mock::MockTransportFactory	2237
activemq::transport::tcp::TcpTransportFactory	2969
activemq::transport::TransportListener	3080
activemq::core::ActiveMQConnectionSupport	220
activemq::core::ActiveMQConnection	202
activemq::transport::DefaultTransportListener	1384
activemq::transport::failover::BackupTransport	591
activemq::transport::mock::InternalCommandListener	1689
activemq::transport::failover::FailoverTransportListener	1525
activemq::transport::TransportFilter	3073
activemq::transport::TransportRegistry	3082
decaf::internal::net::URIEncoderDecoder	3107
decaf::internal::net::URIHelper	3110
activemq::transport::failover::URIPool	3117
activemq::util::URISupport	3119
decaf::internal::net::URIType	3126
decaf::net::URL	3133
decaf::net::URLDecoder	3135
decaf::net::URLEncoder	3136
activemq::util::Usage	3137
activemq::util::MemoryUsage	2014
activemq::wireformat::WireFormat	3148
activemq::wireformat::openwire::OpenWireFormat	2296

activemq::wireformat::stomp::StompWireFormat	2883
activemq::wireformat::WireFormatFactory	3151
activemq::wireformat::openwire::OpenWireFormatFactory	2308
activemq::wireformat::stomp::StompWireFormatFactory	2886
activemq::wireformat::WireFormatRegistry	3183
decaf::io::Writer	3187

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

decaf::util::AbstractCollection < E > (This class provides a skeletal implementation of the Collection (p. 982) interface, to minimize the effort required to implement this interface)	119
decaf::util::AbstractList < E > (This class provides a skeletal implementation of the List (p. 1865) interface to minimize the effort required to implement this interface backed by a "random access" data store (such as an array))	131
decaf::util::AbstractMap < K , V , COMPARATOR > (This class provides a skeletal implementation of the Map (p. 1970) interface, to minimize the effort required to implement this interface)	132
decaf::util::AbstractQueue < E > (This class provides skeletal implementations of some Queue (p. 2507) operations)	133
decaf::util::AbstractSequentialList < E > (This class provides a skeletal implementation of the List (p. 1865) interface to minimize the effort required to implement this interface backed by a "sequential access" data store (such as a linked list))	137
decaf::util::AbstractSet < E > (This class provides a skeletal implementation of the Set (p. 2729) interface to minimize the effort required to implement this interface)	138
activemq::transport::AbstractTransportFactory (Abstract implementation of the TransportFactory (p. 3071) interface, providing the base functionality that's common to most of the TransportFactory (p. 3071) instances)	139
activemq::core::ActiveMQAckHandler (Interface class that is used to give CMS Messages an interface to Ack themselves with)	141
activemq::commands::ActiveMQBlobMessage	142
activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQBlobMessageMarshaller (p. 146))	146
activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQBlobMessageMarshaller (p. 150))	150
activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQBlobMessageMarshaller (p. 154))	154

activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQBlobMessageMarshaller (p. 158))	158
activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQBlobMessageMarshaller (p. 162))	162
activemq::commands::ActiveMQBytesMessage	166
activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQBytesMessageMarshaller (p. 182))	182
activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQBytesMessageMarshaller (p. 186))	186
activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQBytesMessageMarshaller (p. 190))	190
activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQBytesMessageMarshaller (p. 194))	194
activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQBytesMessageMarshaller (p. 198))	198
activemq::core::ActiveMQConnection (Concrete connection used for all connectors to the ActiveMQ broker)	202
activemq::core::ActiveMQConnectionFactory	211
activemq::core::ActiveMQConnectionMetaData (This class houses all the various settings and information that is used by an instance of an ActiveMQConnection (p. 202) class)	216
activemq::core::ActiveMQConnectionSupport	220
activemq::core::ActiveMQConstants (Class holding constant values for various ActiveMQ specific things Each constant is defined as an enumeration and has functions that convert back an forth between string and enum values)	227
activemq::core::ActiveMQConsumer	230
activemq::library::ActiveMQCPP	238
activemq::commands::ActiveMQDestination	240
activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller (Marshaling code for Open Wire Format for ActiveMQDestinationMarshaller (p. 250))	250
activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller (Marshaling code for Open Wire Format for ActiveMQDestinationMarshaller (p. 254))	254
activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller (Marshaling code for Open Wire Format for ActiveMQDestinationMarshaller (p. 258))	258
activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller (Marshaling code for Open Wire Format for ActiveMQDestinationMarshaller (p. 262))	262
activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller (Marshaling code for Open Wire Format for ActiveMQDestinationMarshaller (p. 265))	265
activemq::exceptions::ActiveMQException	269
activemq::commands::ActiveMQMapMessage	272

activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQMapMessageMarshaller (p. 284))	284
activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQMapMessageMarshaller (p. 288))	288
activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQMapMessageMarshaller (p. 292))	292
activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQMapMessageMarshaller (p. 296))	296
activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQMapMessageMarshaller (p. 300))	300
activemq::commands::ActiveMQMessage	304
activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQMessageMarshaller (p. 307))	307
activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQMessageMarshaller (p. 311))	311
activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQMessageMarshaller (p. 315))	315
activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQMessageMarshaller (p. 319))	319
activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQMessageMarshaller (p. 323))	323
activemq::commands::ActiveMQMessageTemplate< T >	327
activemq::commands::ActiveMQObjectMessage	344
activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQObjectMessageMarshaller (p. 347))	347
activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQObjectMessageMarshaller (p. 351))	351
activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQObjectMessageMarshaller (p. 355))	355
activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQObjectMessageMarshaller (p. 359))	359
activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQObjectMessageMarshaller (p. 363))	363
activemq::core::ActiveMQProducer	367
activemq::util::ActiveMQProperties (Implementation of the CMSProperties interface that delegates to a decaf::util::Properties (p. 2494) object)	374
activemq::commands::ActiveMQQueue	379

activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller (Marshaling code for Open Wire Format for ActiveMQQueueMarshaller (p. 382))	382
activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller (Marshaling code for Open Wire Format for ActiveMQQueueMarshaller (p. 386))	386
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller (Marshaling code for Open Wire Format for ActiveMQQueueMarshaller (p. 390))	390
activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller (Marshaling code for Open Wire Format for ActiveMQQueueMarshaller (p. 394))	394
activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller (Marshaling code for Open Wire Format for ActiveMQQueueMarshaller (p. 398))	398
activemq::core::ActiveMQSession	402
activemq::core::ActiveMQSessionExecutor (Delegate dispatcher for a single session)	418
activemq::commands::ActiveMQStreamMessage	422
activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQStreamMessage- Marshaller (p. 436))	436
activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQStreamMessage- Marshaller (p. 439))	439
activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQStreamMessage- Marshaller (p. 443))	443
activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQStreamMessage- Marshaller (p. 447))	447
activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQStreamMessage- Marshaller (p. 451))	451
activemq::commands::ActiveMQTempDestination	455
activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller (Marshaling code for Open Wire Format for ActiveMQTempDestination- Marshaller (p. 459))	459
activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller (Marshaling code for Open Wire Format for ActiveMQTempDestination- Marshaller (p. 462))	462
activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller (Marshaling code for Open Wire Format for ActiveMQTempDestination- Marshaller (p. 466))	466
activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller (Marshaling code for Open Wire Format for ActiveMQTempDestination- Marshaller (p. 470))	470
activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller (Marshaling code for Open Wire Format for ActiveMQTempDestination- Marshaller (p. 473))	473
activemq::commands::ActiveMQTempQueue	477
activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller (Marshaling code for Open Wire Format for ActiveMQTempQueueMar- shaller (p. 481))	481

activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller (Marshaling code for Open Wire Format for ActiveMQTempQueueMarshaller (p. 485))	485
activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller (Marshaling code for Open Wire Format for ActiveMQTempQueueMarshaller (p. 489))	489
activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller (Marshaling code for Open Wire Format for ActiveMQTempQueueMarshaller (p. 493))	493
activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller (Marshaling code for Open Wire Format for ActiveMQTempQueueMarshaller (p. 497))	497
activemq::commands::ActiveMQTempTopic	501
activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller (Marshaling code for Open Wire Format for ActiveMQTempTopicMarshaller (p. 505))	505
activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller (Marshaling code for Open Wire Format for ActiveMQTempTopicMarshaller (p. 509))	509
activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller (Marshaling code for Open Wire Format for ActiveMQTempTopicMarshaller (p. 513))	513
activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller (Marshaling code for Open Wire Format for ActiveMQTempTopicMarshaller (p. 517))	517
activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller (Marshaling code for Open Wire Format for ActiveMQTempTopicMarshaller (p. 521))	521
activemq::commands::ActiveMQTextMessage	525
activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQTextMessageMarshaller (p. 529))	529
activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQTextMessageMarshaller (p. 533))	533
activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQTextMessageMarshaller (p. 537))	537
activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQTextMessageMarshaller (p. 541))	541
activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller (Marshaling code for Open Wire Format for ActiveMQTextMessageMarshaller (p. 545))	545
activemq::commands::ActiveMQTopic	549
activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller (Marshaling code for Open Wire Format for ActiveMQTopicMarshaller (p. 553))	553
activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller (Marshaling code for Open Wire Format for ActiveMQTopicMarshaller (p. 557))	557
activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller (Marshaling code for Open Wire Format for ActiveMQTopicMarshaller (p. 561))	561

activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller (Marshaling code for Open Wire Format for ActiveMQTopicMarshaller (p. 565))	565
activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller (Marshaling code for Open Wire Format for ActiveMQTopicMarshaller (p. 569))	569
activemq::core::ActiveMQTransactionContext (Transaction Management class, hold messages that are to be redelivered upon a request to roll-back)	573
decaf::lang::Appendable	576
decaf::internal::AprPool (Wraps an APR pool object so that classes in decaf can create a static member for use in static methods where apr function calls that need a pool are made)	578
decaf::util::concurrent::atomic::AtomicBoolean (A boolean value that may be up- dated atomically)	579
decaf::util::concurrent::atomic::AtomicInteger (An int value that may be updated atomically)	582
decaf::lang::AtomicRefCounter	587
decaf::util::concurrent::atomic::AtomicReference< T > (An Pointer reference that may be updated atomically)	589
activemq::transport::failover::BackupTransport	591
activemq::transport::failover::BackupTransportPool	594
activemq::commands::BaseCommand	596
activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller (Marshaling code for Open Wire Format for BaseCommandMarshaller (p. 603))	603
activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller (Marshaling code for Open Wire Format for BaseCommandMarshaller (p. 610))	610
activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller (Marshaling code for Open Wire Format for BaseCommandMarshaller (p. 616))	616
activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller (Marshaling code for Open Wire Format for BaseCommandMarshaller (p. 623))	623
activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller (Marshaling code for Open Wire Format for BaseCommandMarshaller (p. 630))	630
activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller (Base class for all Marshallers that marshal DataStructures to and from the wire using the OpenWire protocol)	636
activemq::commands::BaseDataStructure	639
binary_function	663
decaf::net::BindException	663
decaf::io::BlockingByteArrayInputStream (This is a blocking version of a byte buffer stream)	666
decaf::lang::Boolean	674
activemq::commands::BooleanExpression	679
activemq::wireformat::openwire::utils::BooleanStream (Manages the writing and reading of boolean data streams to and from a data source such as a DataInput- Stream or DataOutputStream)	680
decaf::util::concurrent::BrokenBarrierException	683
activemq::commands::BrokerError (This class represents an Exception sent from the Broker)	686
activemq::exceptions::BrokerException	690

activemq::commands::BrokerId	691
activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller (Marshaling code for Open Wire Format for BrokerIdMarshaller (p. 694))	694
activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller (Marshaling code for Open Wire Format for BrokerIdMarshaller (p. 698))	698
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller (Marshaling code for Open Wire Format for BrokerIdMarshaller (p. 701))	701
activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller (Marshaling code for Open Wire Format for BrokerIdMarshaller (p. 705))	705
activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller (Marshaling code for Open Wire Format for BrokerIdMarshaller (p. 709))	709
activemq::commands::BrokerInfo	713
activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller (Marshaling code for Open Wire Format for BrokerInfoMarshaller (p. 721))	721
activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller (Marshaling code for Open Wire Format for BrokerInfoMarshaller (p. 725))	725
activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller (Marshaling code for Open Wire Format for BrokerInfoMarshaller (p. 729))	729
activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller (Marshaling code for Open Wire Format for BrokerInfoMarshaller (p. 733))	733
activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller (Marshaling code for Open Wire Format for BrokerInfoMarshaller (p. 737))	737
decaf::nio::Buffer (A container for data of a specific primitive type)	741
decaf::io::BufferedInputStream (A wrapper around another input stream that performs a buffered read, where it reads more data than it needs in order to reduce the number of io operations on the input stream)	747
decaf::io::BufferedOutputStream (Wrapper around another output stream that buffers output before writing to the target output stream)	752
decaf::net::BufferedSocket (Buffered Socket (p. 2786) class that wraps a Socket (p. 2786) derived object and provides Buffered input and Output Streams to improve the efficiency of the reads and writes)	755
decaf::internal::nio::BufferFactory (Factory class used by static methods in the decaf::nio (p. 109) package to create the various default version of the NIO interfaces)	762
decaf::nio::BufferOverflowException	771
decaf::nio::BufferUnderflowException	774
decaf::lang::Byte	776
decaf::internal::util::ByteArrayAdapter (This class adapts primitive type arrays to a base byte array so that the classes can inter-operate on the same base byte array without copying data)	784
decaf::internal::nio::ByteArrayBuffer (This class defines six categories of operations upon byte buffers:)	806
decaf::io::ByteArrayInputStream (Simple implementation of InputStream (p. 1630) that wraps around an STL Vector <code>std::vector<unsigned char></code>)	828
decaf::io::ByteArrayOutputStream	836
decaf::internal::nio::ByteArrayPerspective (This class extends ByteArray to create a reference counted byte array that can be held and used by several different ByteBuffers and allow them to know on destruction whose job it is to delete the perspective)	843
decaf::nio::ByteBuffer (This class defines six categories of operations upon byte buffers:)	848
cms::BytesMessage (A BytesMessage (p. 874) object is used to send a message containing a stream of unsigned bytes)	874

activemq::cmsutil::CachedConsumer (A cached message consumer contained within a pooled session)	888
activemq::cmsutil::CachedProducer (A cached message producer contained within a pooled session)	891
decaf::util::concurrent::Callable< V > (A task that returns a result and may throw an exception)	897
decaf::util::concurrent::CancellationException	898
decaf::security::cert::Certificate (Base interface for all identity certificates)	901
decaf::security::cert::CertificateEncodingException	904
decaf::security::cert::CertificateException	906
decaf::security::cert::CertificateExpiredException	908
decaf::security::cert::CertificateNotYetValidException	910
decaf::security::cert::CertificateParsingException	912
decaf::lang::Character	914
decaf::internal::nio::CharArrayBuffer	922
decaf::nio::CharBuffer (This class defines four categories of operations upon character buffers:)	930
decaf::lang::CharSequence (A CharSequence (p. 946) is a readable sequence of char values)	946
decaf::lang::exceptions::ClassCastException	948
decaf::io::Closeable (Interface for a class that implements the close method)	950
cms::Closeable (Interface for a class that implements the close method)	951
activemq::transport::failover::CloseTransportsTask	952
activemq::cmsutil::CmsAccessor (Base class for activemq::cmsutil::CmsTemplate (p. 969) and other CMS-accessing gateway helpers, defining common properties such as the CMS cms::ConnectionFactory (p. 1103) to operate on)	954
activemq::cmsutil::CmsDestinationAccessor (Extends the CmsAccessor (p. 954) to add support for resolving destination names)	957
cms::CMSException (CMS API Exception that is the base for all exceptions thrown from CMS classes)	960
activemq::util::CMSExceptionSupport	963
cms::CMSProperties (Interface for a Java-like properties object)	965
cms::CMSSecurityException (This exception must be thrown when a provider rejects a user name/password submitted by a client)	968
activemq::cmsutil::CmsTemplate (CmsTemplate (p. 969) simplifies performing synchronous CMS operations)	969
decaf::util::Collection< E > (The root interface in the collection hierarchy)	982
activemq::commands::Command	991
activemq::state::CommandVisitor (Interface for an Object that can visit the various Command Objects that are sent from and to this client)	996
activemq::state::CommandVisitorAdapter (Default Implementation of a CommandVisitor (p. 996) that returns NULL for all calls)	1003
decaf::lang::Comparable< T > (This interface imposes a total ordering on the objects of each class that implements it)	1009
decaf::util::Comparator< T > (A comparison function, which imposes a total ordering on some collection of objects)	1011
activemq::util::CompositeData (Represents a Composite URI)	1013
activemq::threads::CompositeTask (Represents a single task that can be part of a set of Tasks that are contained in a CompositeTaskRunner (p. 1017))	1016
activemq::threads::CompositeTaskRunner (A Task (p. 2956) Runner that can contain one or more CompositeTasks that are each checked for pending work and run if any is present in the order that the tasks were added)	1017
activemq::transport::CompositeTransport (A Composite Transport (p. 3066) is a Transport (p. 3066) implementation that is composed of several Transports)	1019

decaf::util::concurrent::ConcurrentMap < K , V , COMPARATOR > (Interface for a Map (p. 1970) type that provides additional atomic putIfAbsent , remove , and replace methods alongside the already available Map (p. 1970) interface) .	1020
decaf::util::concurrent::ConcurrentStlMap < K , V , COMPARATOR > (Map (p. 1970) template that wraps around a std::map to provide a more user-friendly interface and to provide common functions that do not exist in std::map) . . .	1024
decaf::util::concurrent::locks::Condition (Condition (p. 1040) factors out the Mutex (p. 2239) monitor methods (wait , notify and notifyAll) into distinct objects to give the effect of having multiple wait-sets per object, by combining them with the use of arbitrary Lock (p. 1898) implementations)	1040
decaf::util::concurrent::ConditionHandle	1046
decaf::internal::util::concurrent::ConditionImpl	1047
decaf::net::ConnectException	1049
cms::Connection (The client's connection to its provider)	1052
activemq::commands::ConnectionControl	1055
activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller (Marshaling code for Open Wire Format for ConnectionControlMarshaller (p. 1059))	1059
activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller (Marshaling code for Open Wire Format for ConnectionControlMarshaller (p. 1063))	1063
activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller (Marshaling code for Open Wire Format for ConnectionControlMarshaller (p. 1067))	1067
activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller (Marshaling code for Open Wire Format for ConnectionControlMarshaller (p. 1071))	1071
activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller (Marshaling code for Open Wire Format for ConnectionControlMarshaller (p. 1075))	1075
activemq::commands::ConnectionError	1079
activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller (Marshaling code for Open Wire Format for ConnectionErrorMarshaller (p. 1083))	1083
activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller (Marshaling code for Open Wire Format for ConnectionErrorMarshaller (p. 1087))	1087
activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller (Marshaling code for Open Wire Format for ConnectionErrorMarshaller (p. 1091))	1091
activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller (Marshaling code for Open Wire Format for ConnectionErrorMarshaller (p. 1095))	1095
activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller (Marshaling code for Open Wire Format for ConnectionErrorMarshaller (p. 1099))	1099
cms::ConnectionFactory (Defines the interface for a factory that creates connection objects, the Connection (p. 1052) objects returned implement the CMS Connection (p. 1052) interface and hide the CMS Provider specific implementation details behind that interface)	1103
activemq::commands::ConnectionId	1106
activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller (Marshaling code for Open Wire Format for ConnectionIdMarshaller (p. 1109))	1109

activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller (Marshaling code for Open Wire Format for ConnectionIdMarshaller (p. 1113))	1113
activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller (Marshaling code for Open Wire Format for ConnectionIdMarshaller (p. 1116))	1116
activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller (Marshaling code for Open Wire Format for ConnectionIdMarshaller (p. 1120))	1120
activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller (Marshaling code for Open Wire Format for ConnectionIdMarshaller (p. 1124))	1124
activemq::commands::ConnectionInfo	1128
activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller (Marshaling code for Open Wire Format for ConnectionInfoMarshaller (p. 1134))	1134
activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller (Marshaling code for Open Wire Format for ConnectionInfoMarshaller (p. 1138))	1138
activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller (Marshaling code for Open Wire Format for ConnectionInfoMarshaller (p. 1142))	1142
activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller (Marshaling code for Open Wire Format for ConnectionInfoMarshaller (p. 1146))	1146
activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller (Marshaling code for Open Wire Format for ConnectionInfoMarshaller (p. 1150))	1150
cms::ConnectionMetaData (A ConnectionMetaData (p. 1154) object provides information describing the Connection (p. 1052) object)	1154
activemq::state::ConnectionState	1158
activemq::state::ConnectionStateTracker	1160
activemq::commands::ConsumerControl	1165
activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller (Marshaling code for Open Wire Format for ConsumerControlMarshaller (p. 1170))	1170
activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller (Marshaling code for Open Wire Format for ConsumerControlMarshaller (p. 1174))	1174
activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller (Marshaling code for Open Wire Format for ConsumerControlMarshaller (p. 1178))	1178
activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller (Marshaling code for Open Wire Format for ConsumerControlMarshaller (p. 1182))	1182
activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller (Marshaling code for Open Wire Format for ConsumerControlMarshaller (p. 1186))	1186
activemq::commands::ConsumerId	1190
activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller (Marshaling code for Open Wire Format for ConsumerIdMarshaller (p. 1195))	1195

activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller (Marshaling code for Open Wire Format for ConsumerIdMarshaller (p. 1198))	1198
activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller (Marshaling code for Open Wire Format for ConsumerIdMarshaller (p. 1202))	1202
activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller (Marshaling code for Open Wire Format for ConsumerIdMarshaller (p. 1206))	1206
activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller (Marshaling code for Open Wire Format for ConsumerIdMarshaller (p. 1210))	1210
activemq::commands::ConsumerInfo	1214
activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller (Marshaling code for Open Wire Format for ConsumerInfoMarshaller (p. 1223))	1223
activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller (Marshaling code for Open Wire Format for ConsumerInfoMarshaller (p. 1226))	1226
activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller (Marshaling code for Open Wire Format for ConsumerInfoMarshaller (p. 1230))	1230
activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller (Marshaling code for Open Wire Format for ConsumerInfoMarshaller (p. 1234))	1234
activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller (Marshaling code for Open Wire Format for ConsumerInfoMarshaller (p. 1238))	1238
activemq::state::ConsumerState	1242
activemq::commands::ControlCommand	1243
activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller (Marshaling code for Open Wire Format for ControlCommandMarshaller (p. 1246))	1246
activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller (Marshaling code for Open Wire Format for ControlCommandMarshaller (p. 1250))	1250
activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller (Marshaling code for Open Wire Format for ControlCommandMarshaller (p. 1254))	1254
activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller (Marshaling code for Open Wire Format for ControlCommandMarshaller (p. 1258))	1258
activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller (Marshaling code for Open Wire Format for ControlCommandMarshaller (p. 1262))	1262
decaf::util::concurrent::CountDownLatch	1266
activemq::commands::DataArrayResponse	1267
activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller (Marshaling code for Open Wire Format for DataArrayResponseMarshaller (p. 1271))	1271
activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller (Marshaling code for Open Wire Format for DataArrayResponseMarshaller (p. 1275))	1275

activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller (Marshaling code for Open Wire Format for DataArrayResponseMarshaller (p. 1279))	1279
activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller (Marshaling code for Open Wire Format for DataArrayResponseMarshaller (p. 1283))	1283
activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller (Marshaling code for Open Wire Format for DataArrayResponseMarshaller (p. 1287))	1287
decaf::io::DataInputStream (A data input stream lets an application read primitive Java data types from an underlying input stream in a machine-independent way)	1291
decaf::io::DataOutputStream (A data output stream lets an application write prim- itive Java data types to an output stream in a portable way)	1300
activemq::commands::DataResponse	1307
activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller (Mar- shaling code for Open Wire Format for DataResponseMarshaller (p. 1310))	1310
activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller (Mar- shaling code for Open Wire Format for DataResponseMarshaller (p. 1314))	1314
activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller (Mar- shaling code for Open Wire Format for DataResponseMarshaller (p. 1318))	1318
activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller (Mar- shaling code for Open Wire Format for DataResponseMarshaller (p. 1322))	1322
activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller (Mar- shaling code for Open Wire Format for DataResponseMarshaller (p. 1326))	1326
activemq::wireformat::openwire::marshal::DataStreamMarshaller (Base class for all classes that marshal commands for Openwire)	1330
activemq::commands::DataStructure	1372
decaf::util::Date (Wrapper class around a time value in milliseconds)	1377
decaf::internal::DecafRuntime (Handles APR initialization and termination)	1381
activemq::threads::DedicatedTaskRunner	1382
activemq::transport::DefaultTransportListener	1384
decaf::util::concurrent::Delayed (A mix-in style interface for marking objects that should be acted upon after a given delay)	1385
cms::DeliveryMode (This is an Abstract class whose purpose is to provide a container for the delivery mode enumeration for CMS messages)	1386
cms::Destination (A Destination (p. 1387) object encapsulates a provider-specific ad- dress)	1387
activemq::commands::ActiveMQDestination::DestinationFilter	1390
activemq::commands::DestinationInfo	1390
activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller (Marshaling code for Open Wire Format for DestinationInfoMarshaller (p. 1395))	1395
activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller (Marshaling code for Open Wire Format for DestinationInfoMarshaller (p. 1399))	1399
activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller (Marshaling code for Open Wire Format for DestinationInfoMarshaller (p. 1403))	1403

activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller (Marshaling code for Open Wire Format for DestinationInfoMarshaller (p. 1407))	1407
activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller (Marshaling code for Open Wire Format for DestinationInfoMarshaller (p. 1411))	1411
activemq::cmsutil::DestinationResolver (Resolves a CMS destination name to a Destination)	1415
activemq::commands::DiscoveryEvent	1416
activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller (Marshaling code for Open Wire Format for DiscoveryEventMarshaller (p. 1420))	1420
activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller (Marshaling code for Open Wire Format for DiscoveryEventMarshaller (p. 1424))	1424
activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller (Marshaling code for Open Wire Format for DiscoveryEventMarshaller (p. 1427))	1427
activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller (Marshaling code for Open Wire Format for DiscoveryEventMarshaller (p. 1431))	1431
activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller (Marshaling code for Open Wire Format for DiscoveryEventMarshaller (p. 1435))	1435
activemq::core::DispatchData (Simple POJO that contains the information neces- sary to route a message to a specified consumer)	1439
activemq::core::Dispatcher (Interface for an object responsible for dispatching mes- sages to consumers)	1440
decaf::lang::Double	1441
decaf::internal::nio::DoubleArrayBuffer	1452
decaf::nio::DoubleBuffer (This class defines four categories of operations upon double buffers:)	1459
decaf::lang::DYNAMIC_CAST_TOKEN	1471
activemq::cmsutil::DynamicDestinationResolver (Resolves a CMS destination name to a Destination)	1471
decaf::util::Map< K, V, COMPARATOR >::Entry	1473
decaf::io::EOFException	1474
decaf::lang::Exception	1476
cms::ExceptionListener (If a CMS provider detects a serious problem, it notifies the client application through an ExceptionListener (p. 1483) that is registered with the Connection (p. 1052))	1483
activemq::commands::ExceptionResponse	1484
activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller (Marshaling code for Open Wire Format for ExceptionResponseMarshaller (p. 1487))	1487
activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller (Marshaling code for Open Wire Format for ExceptionResponseMarshaller (p. 1491))	1491
activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller (Marshaling code for Open Wire Format for ExceptionResponseMarshaller (p. 1495))	1495
activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller (Marshaling code for Open Wire Format for ExceptionResponseMarshaller (p. 1499))	1499

activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller (Marshaling code for Open Wire Format for ExceptionResponseMarshaller (p. 1503))	1503
decaf::util::concurrent::ExecutionException	1507
decaf::util::concurrent::Executor (An object that executes submitted decaf.lang Runnable (p. 2642) tasks)	1509
decaf::util::concurrent::ExecutorService (An Executor (p. 1509) that provides methods to manage termination and methods that can produce a Future (p. 1597) for tracking progress of one or more asynchronous tasks)	1511
activemq::transport::failover::FailoverTransport	1512
activemq::transport::failover::FailoverTransportFactory (Creates an instance of a FailoverTransport (p. 1512))	1523
activemq::transport::failover::FailoverTransportListener (Utility class used by the Transport (p. 3066) to perform the work of responding to events from the active Transport (p. 3066))	1525
decaf::util::logging::Filter (A Filter (p. 1527) can be used to provide fine grain control over what is logged, beyond the control provided by log levels)	1527
decaf::io::FilterInputStream (A FilterInputStream (p. 1528) contains some other input stream, which it uses as its basic source of data, possibly transforming the data along the way or providing additional functionality)	1528
decaf::io::FilterOutputStream (This class is the superclass of all classes that filter output streams)	1536
decaf::lang::Float	1544
decaf::internal::nio::FloatArrayBuffer	1555
decaf::nio::FloatBuffer (This class defines four categories of operations upon float buffers:)	1562
activemq::commands::FlushCommand	1573
activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller (Marshaling code for Open Wire Format for FlushCommandMarshaller (p. 1576))	1576
activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller (Marshaling code for Open Wire Format for FlushCommandMarshaller (p. 1579))	1579
activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller (Marshaling code for Open Wire Format for FlushCommandMarshaller (p. 1583))	1583
activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller (Marshaling code for Open Wire Format for FlushCommandMarshaller (p. 1587))	1587
activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller (Marshaling code for Open Wire Format for FlushCommandMarshaller (p. 1591))	1591
decaf::util::logging::Formatter (A Formatter (p. 1595) provides support for formatting LogRecords)	1595
decaf::util::concurrent::Future< V > (A Future (p. 1597) represents the result of an asynchronous computation)	1597
activemq::transport::correlator::FutureResponse (A container that holds a response object)	1600
decaf::security::GeneralSecurityException	1602
decaf::util::logging::Handler (A Handler (p. 1604) object takes log messages from a Logger (p. 1908) and exports them)	1604
decaf::internal::util::HexStringParser	1607
activemq::wireformat::openwire::utils::HexTable (Maps hexadecimal strings to the value of an index into the table, i.e)	1609

decaf::net::HttpRetryException	1610
decaf::lang::exceptions::IllegalArgumentException	1613
decaf::lang::exceptions::IllegalMonitorStateException	1616
decaf::lang::exceptions::IllegalStateException	1618
cms::IllegalStateException (This exception is thrown when a method is invoked at an illegal or inappropriate time or if the provider is not in an appropriate state for the requested operation)	1621
decaf::lang::exceptions::IllegalThreadStateException	1622
activemq::transport::inactivity::InactivityMonitor	1624
decaf::lang::exceptions::IndexOutOfBoundsException	1627
decaf::io::InputStream (Base interface for an input stream)	1630
decaf::internal::nio::IntArrayBuffer	1634
decaf::nio::IntBuffer (This class defines four categories of operations upon int buffers:)	1641
decaf::lang::Integer	1652
activemq::commands::IntegerResponse	1667
activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller (Marshaling code for Open Wire Format for IntegerResponseMarshaller (p. 1669))	1669
activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller (Marshaling code for Open Wire Format for IntegerResponseMarshaller (p. 1673))	1673
activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller (Marshaling code for Open Wire Format for IntegerResponseMarshaller (p. 1677))	1677
activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller (Marshaling code for Open Wire Format for IntegerResponseMarshaller (p. 1681))	1681
activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller (Marshaling code for Open Wire Format for IntegerResponseMarshaller (p. 1685))	1685
activemq::transport::mock::InternalCommandListener (Listens for Commands sent from the MockTransport (p. 2227))	1689
decaf::lang::exceptions::InterruptedException	1691
decaf::io::InterruptedIOException	1693
cms::InvalidClientIdException (This exception must be thrown when a client attempts to set a connection's client ID to a value that is rejected by a provider)	1696
cms::InvalidDestinationException (This exception must be thrown when a destination either is not understood by a provider or is no longer valid)	1697
decaf::security::InvalidKeyException	1698
decaf::nio::InvalidMarkException	1700
cms::InvalidSelectorException (This exception must be thrown when a CMS client attempts to give a provider a message selector with invalid syntax)	1703
decaf::lang::exceptions::InvalidStateException	1704
decaf::io::IOException	1707
activemq::transport::IOTransport (Implementation of the Transport (p. 3066) interface that performs marshaling of commands to IO streams)	1709
decaf::lang::Iterable< E > (Implementing this interface allows an object to be cast to an Iterable (p. 1715) type for generic collections API calls)	1715
decaf::util::Iterator< T > (Defines an object that can be used to iterate over the elements of a collection)	1716
activemq::commands::JournalQueueAck	1718

activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller (Marshaling code for Open Wire Format for JournalQueueAckMarshaller (p. 1722))	1722
activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller (Marshaling code for Open Wire Format for JournalQueueAckMarshaller (p. 1725))	1725
activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller (Marshaling code for Open Wire Format for JournalQueueAckMarshaller (p. 1729))	1729
activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller (Marshaling code for Open Wire Format for JournalQueueAckMarshaller (p. 1733))	1733
activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller (Marshaling code for Open Wire Format for JournalQueueAckMarshaller (p. 1737))	1737
activemq::commands::JournalTopicAck	1741
activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller (Marshaling code for Open Wire Format for JournalTopicAckMarshaller (p. 1746))	1746
activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller (Marshaling code for Open Wire Format for JournalTopicAckMarshaller (p. 1750))	1750
activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller (Marshaling code for Open Wire Format for JournalTopicAckMarshaller (p. 1754))	1754
activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller (Marshaling code for Open Wire Format for JournalTopicAckMarshaller (p. 1758))	1758
activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller (Marshaling code for Open Wire Format for JournalTopicAckMarshaller (p. 1762))	1762
activemq::commands::JournalTrace	1766
activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller (Mar- shaling code for Open Wire Format for JournalTraceMarshaller (p. 1769))	1769
activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller (Mar- shaling code for Open Wire Format for JournalTraceMarshaller (p. 1773))	1773
activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller (Mar- shaling code for Open Wire Format for JournalTraceMarshaller (p. 1777))	1777
activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller (Mar- shaling code for Open Wire Format for JournalTraceMarshaller (p. 1781))	1781
activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller (Mar- shaling code for Open Wire Format for JournalTraceMarshaller (p. 1785))	1785
activemq::commands::JournalTransaction	1789
activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller (Marshaling code for Open Wire Format for JournalTransactionMarshaller (p. 1793))	1793
activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller (Marshaling code for Open Wire Format for JournalTransactionMarshaller (p. 1797))	1797

activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller (Marshaling code for Open Wire Format for JournalTransactionMarshaller (p. 1801))	1801
activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller (Marshaling code for Open Wire Format for JournalTransactionMarshaller (p. 1804))	1804
activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller (Marshaling code for Open Wire Format for JournalTransactionMarshaller (p. 1808))	1808
activemq::commands::KeepAliveInfo	1812
activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller (Mar- shaling code for Open Wire Format for KeepAliveInfoMarshaller (p. 1815))	1815
activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller (Mar- shaling code for Open Wire Format for KeepAliveInfoMarshaller (p. 1819))	1819
activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller (Mar- shaling code for Open Wire Format for KeepAliveInfoMarshaller (p. 1823))	1823
activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller (Mar- shaling code for Open Wire Format for KeepAliveInfoMarshaller (p. 1827))	1827
activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller (Mar- shaling code for Open Wire Format for KeepAliveInfoMarshaller (p. 1831))	1831
decaf::security::Key (The Key (p. 1835) interface is the top-level interface for all keys)	1835
decaf::security::KeyException	1837
activemq::commands::LastPartialCommand	1840
activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller (Marshaling code for Open Wire Format for LastPartialCommandMar- shaller (p. 1842))	1842
activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller (Marshaling code for Open Wire Format for LastPartialCommandMar- shaller (p. 1846))	1846
activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller (Marshaling code for Open Wire Format for LastPartialCommandMar- shaller (p. 1850))	1850
activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller (Marshaling code for Open Wire Format for LastPartialCommandMar- shaller (p. 1854))	1854
activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller (Marshaling code for Open Wire Format for LastPartialCommandMar- shaller (p. 1858))	1858
decaf::util::comparators::Less< E > (Simple Less (p. 1862) Comparator (p. 1011) that compares to elements to determine if the first is less than the second) . .	1862
std::less< decaf::lang::Pointer< T > > (An override of the less function object so that the Pointer objects can be stored in STL Maps, etc)	1864
decaf::util::List< E > (An ordered collection (also known as a sequence))	1865
decaf::util::ListIterator< E > (An iterator for lists that allows the programmer to traverse the list in either direction, modify the list during iteration, and obtain the iterator's current position in the list)	1871
activemq::commands::LocalTransactionId	1874

activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller (Marshaling code for Open Wire Format for LocalTransactionIdMarshaller (p. 1878))	1878
activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller (Marshaling code for Open Wire Format for LocalTransactionIdMarshaller (p. 1882))	1882
activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller (Marshaling code for Open Wire Format for LocalTransactionIdMarshaller (p. 1886))	1886
activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller (Marshaling code for Open Wire Format for LocalTransactionIdMarshaller (p. 1890))	1890
activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller (Marshaling code for Open Wire Format for LocalTransactionIdMarshaller (p. 1894))	1894
decaf::util::concurrent::locks::Lock (Lock (p. 1898) implementations provide more extensive locking operations than can be obtained using synchronized statements)	1898
decaf::util::concurrent::Lock (A wrapper class around a given synchronization mech- anism that provides automatic release upon destruction)	1903
decaf::util::concurrent::locks::LockSupport (Basic thread blocking primitives for creating locks and other synchronization classes)	1905
decaf::util::logging::Logger	1908
decaf::util::logging::LoggerHierarchy	1916
activemq::io::LoggingInputStream	1917
activemq::io::LoggingOutputStream (OutputStream filter that just logs the data being written)	1919
activemq::transport::logging::LoggingTransport (A transport filter that logs com- mands as they are sent/received)	1920
decaf::util::logging::LogManager (There is a single global LogManager (p. 1923) object that is used to maintain a set of shared state about Loggers and log services)	1923
decaf::util::logging::LogRecord	1928
decaf::util::logging::LogWriter	1932
decaf::lang::Long	1934
decaf::internal::nio::LongArrayBuffer	1948
decaf::nio::LongBuffer (This class defines four categories of operations upon long long buffers:)	1955
activemq::util::LongSequenceGenerator (This class is used to generate a sequence of long long values that are incremented each time a new value is requested) .	1967
decaf::net::MalformedURLException	1968
decaf::util::Map< K, V, COMPARATOR > (Map (p. 1970) template that wraps around a std::map to provide a more user-friendly interface and to provide com- mon functions that do not exist in std::map)	1970
cms::MapMessage (A MapMessage (p. 1982) object is used to send a set of name- value pairs)	1982
decaf::util::logging::MarkBlockLogger (Defines a class that can be used to mark the entry and exit from scoped blocks)	1992
activemq::wireformat::MarshalAware	1992
activemq::wireformat::openwire::marshal::v2::MarshallerFactory (Used to cre- ate marshallers for a specific version of the wire protocol)	1995
activemq::wireformat::openwire::marshal::v4::MarshallerFactory (Used to cre- ate marshallers for a specific version of the wire protocol)	1996
activemq::wireformat::openwire::marshal::v1::MarshallerFactory (Used to cre- ate marshallers for a specific version of the wire protocol)	1997

activemq::wireformat::openwire::marshal::v3::MarshallerFactory (Used to create marshallers for a specific version of the wire protocol)	1997
activemq::wireformat::openwire::marshal::v5::MarshallerFactory (Used to create marshallers for a specific version of the wire protocol)	1998
decaf::lang::Math (The class Math (p. 1999) contains methods for performing basic numeric operations such as the elementary exponential, logarithm, square root, and trigonometric functions)	1999
activemq::util::MemoryUsage	2014
activemq::commands::Message	2018
cms::Message (Root of all messages)	2036
activemq::commands::MessageAck	2055
activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller (Mar- shaling code for Open Wire Format for MessageAckMarshaller (p. 2060))	2060
activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller (Mar- shaling code for Open Wire Format for MessageAckMarshaller (p. 2064))	2064
activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller (Mar- shaling code for Open Wire Format for MessageAckMarshaller (p. 2068))	2068
activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller (Mar- shaling code for Open Wire Format for MessageAckMarshaller (p. 2072))	2072
activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller (Mar- shaling code for Open Wire Format for MessageAckMarshaller (p. 2076))	2076
cms::MessageConsumer (A client uses a MessageConsumer (p. 2080) to received mes- sages from a destination)	2080
activemq::cmsutil::MessageCreator (Creates the user-defined message to be sent by the CmsTemplate (p. 969))	2083
activemq::commands::MessageDispatch	2084
activemq::core::MessageDispatchChannel	2088
activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller (Marshaling code for Open Wire Format for MessageDispatchMarshaller (p. 2095))	2095
activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller (Marshaling code for Open Wire Format for MessageDispatchMarshaller (p. 2099))	2099
activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller (Marshaling code for Open Wire Format for MessageDispatchMarshaller (p. 2103))	2103
activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller (Marshaling code for Open Wire Format for MessageDispatchMarshaller (p. 2107))	2107
activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller (Marshaling code for Open Wire Format for MessageDispatchMarshaller (p. 2111))	2111
activemq::commands::MessageDispatchNotification	2115
activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller (Marshaling code for Open Wire Format for MessageDispatchNotification- Marshaller (p. 2120))	2120
activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller (Marshaling code for Open Wire Format for MessageDispatchNotification- Marshaller (p. 2124))	2124

activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller (Marshaling code for Open Wire Format for MessageDispatchNotificationMarshaller (p. 2128))	2128
activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller (Marshaling code for Open Wire Format for MessageDispatchNotificationMarshaller (p. 2132))	2132
activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller (Marshaling code for Open Wire Format for MessageDispatchNotificationMarshaller (p. 2136))	2136
cms::MessageEOFException (This exception must be thrown when an unexpected end of stream has been reached when a StreamMessage (p. 2892) or BytesMessage (p. 874) is being read)	2140
cms::MessageFormatException (This exception must be thrown when a CMS client attempts to use a data type not supported by a message or attempts to read data in a message as the wrong type)	2141
activemq::commands::MessageId	2142
activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller (Marshaling code for Open Wire Format for MessageIdMarshaller (p. 2146))	2146
activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller (Marshaling code for Open Wire Format for MessageIdMarshaller (p. 2150))	2150
activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller (Marshaling code for Open Wire Format for MessageIdMarshaller (p. 2154))	2154
activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller (Marshaling code for Open Wire Format for MessageIdMarshaller (p. 2157))	2157
activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller (Marshaling code for Open Wire Format for MessageIdMarshaller (p. 2161))	2161
cms::MessageListener (A MessageListener (p. 2165) object is used to receive asynchronously delivered messages)	2165
activemq::wireformat::openwire::marshal::v2::MessageMarshaller (Marshaling code for Open Wire Format for MessageMarshaller (p. 2166))	2166
activemq::wireformat::openwire::marshal::v4::MessageMarshaller (Marshaling code for Open Wire Format for MessageMarshaller (p. 2170))	2170
activemq::wireformat::openwire::marshal::v3::MessageMarshaller (Marshaling code for Open Wire Format for MessageMarshaller (p. 2175))	2175
activemq::wireformat::openwire::marshal::v5::MessageMarshaller (Marshaling code for Open Wire Format for MessageMarshaller (p. 2179))	2179
activemq::wireformat::openwire::marshal::v1::MessageMarshaller (Marshaling code for Open Wire Format for MessageMarshaller (p. 2183))	2183
cms::MessageNotReadableException (This exception must be thrown when a CMS client attempts to read a write-only message)	2187
cms::MessageNotWriteableException (This exception must be thrown when a CMS client attempts to write to a read-only message)	2188
cms::MessageProducer (A client uses a MessageProducer (p. 2189) object to send messages to a Destination (p. 1387))	2189
activemq::wireformat::openwire::utils::MessagePropertyInterceptor (Used the base ActiveMQMessage class to intercept calls to get and set properties in order to capture the calls that use the reserved JMS properties and get and set them in the OpenWire Message properties)	2196
activemq::commands::MessagePull	2202
activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller (Marshaling code for Open Wire Format for MessagePullMarshaller (p. 2207))	2207

activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller (Marshaling code for Open Wire Format for MessagePullMarshaller (p. 2211))	2211
activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller (Marshaling code for Open Wire Format for MessagePullMarshaller (p. 2215))	2215
activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller (Marshaling code for Open Wire Format for MessagePullMarshaller (p. 2219))	2219
activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller (Marshaling code for Open Wire Format for MessagePullMarshaller (p. 2223))	2223
activemq::transport::mock::MockTransport (The MockTransport (p. 2227) defines a base level Transport (p. 3066) class that is intended to be used in place of an a regular protocol Transport (p. 3066) such as TCP)	2227
activemq::transport::mock::MockTransportFactory (Manufactures MockTransports , which are objects that read from input streams and write to output streams)	2237
decaf::util::concurrent::Mutex (Mutex (p. 2239) object that offers recursive support on all platforms as well as providing the ability to use the standard wait / notify pattern used in languages like Java)	2239
decaf::util::concurrent::MutexHandle	2244
decaf::internal::util::concurrent::MutexImpl	2244
activemq::commands::NetworkBridgeFilter	2246
activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller (Marshaling code for Open Wire Format for NetworkBridgeFilterMarshaller (p. 2250))	2250
activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller (Marshaling code for Open Wire Format for NetworkBridgeFilterMarshaller (p. 2253))	2253
activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller (Marshaling code for Open Wire Format for NetworkBridgeFilterMarshaller (p. 2257))	2257
activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller (Marshaling code for Open Wire Format for NetworkBridgeFilterMarshaller (p. 2261))	2261
activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller (Marshaling code for Open Wire Format for NetworkBridgeFilterMarshaller (p. 2265))	2265
decaf::net::NoRouteToHostException	2269
decaf::security::NoSuchAlgorithmException	2272
decaf::lang::exceptions::NoSuchElementException	2274
decaf::security::NoSuchProviderException	2277
decaf::lang::exceptions::NullPointerException	2279
decaf::lang::Number (The abstract class Number (p. 2282) is the superclass of classes Byte (p. 776), Double (p. 1441), Float (p. 1544), Integer (p. 1652), Long (p. 1934), and Short (p. 2729))	2282
decaf::lang::exceptions::NumberFormatException	2284
cms::ObjectMessage (Place holder for interaction with JMS systems that support Java, the C++ client is not responsible for deserializing the contained Object)	2287
decaf::security_provider::unix::openssl::OpenSSLX500Principal (The OpenSSLX500Principal (p. 2287) wraps around an OpenSSL X509_NAME structure)	2287
decaf::security_provider::unix::openssl::OpenSSLX509Certificate	2290

activemq::wireformat::openwire::OpenWireFormat	2296
activemq::wireformat::openwire::OpenWireFormatFactory	2308
activemq::wireformat::openwire::OpenWireFormatNegotiator	2310
activemq::wireformat::openwire::OpenWireResponseBuilder	2313
activemq::wireformat::openwire::utils::OpenwireStringSupport	2315
decaf::io::OutputStream (Base interface for an output stream)	2316
activemq::commands::PartialCommand	2318
activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller (Marshaling code for Open Wire Format for PartialCommandMarshaller (p. 2322))	2322
activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller (Marshaling code for Open Wire Format for PartialCommandMarshaller (p. 2326))	2326
activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller (Marshaling code for Open Wire Format for PartialCommandMarshaller (p. 2330))	2330
activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller (Marshaling code for Open Wire Format for PartialCommandMarshaller (p. 2335))	2335
activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller (Marshaling code for Open Wire Format for PartialCommandMarshaller (p. 2339))	2339
decaf::lang::Pointer< T, REFCOUNTER > (Decaf's implementation of a Smart Pointer (p. 2343) that is a template on a Type and is Thread Safe if the default Reference Counter is used)	2343
decaf::lang::PointerComparator< T, R > (This implementation of Comparator is designed to allows objects in a Collection to be sorted or tested for equality based on the value of the Object being Pointed to and not the value of the contained pointer in the Pointer (p. 2343) instance)	2350
activemq::cmsutil::PooledSession (A pooled session object that wraps around a del- egate session)	2351
decaf::util::concurrent::PooledThread	2364
decaf::util::concurrent::PooledThreadListener (Abstract Listener Interface for users of ThreadPool (p. 2977))	2366
decaf::net::PortUnreachableException	2368
activemq::util::PrimitiveList (List of primitives)	2370
activemq::util::PrimitiveMap (Map of named primitives)	2381
activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller (This class wraps the functionality needed to marshal a primitive map to the Openwire Format's expectation of what the map looks like on the wire)	2390
activemq::util::PrimitiveValueNode::PrimitiveValue (Define a union type com- prised of the various types)	2395
activemq::util::PrimitiveValueConverter (Class controls the conversion of data con- tained in a PrimitiveValueNode (p. 2398) from one type to another)	2397
activemq::util::PrimitiveValueNode (Class that wraps around a single value of one of the many types)	2398
decaf::security::Principal (Base interface for a principal, which can represent an indi- vidual or organization)	2411
decaf::util::PriorityQueue< E > (An unbounded priority queue based on a binary heap algorithm)	2413
activemq::commands::ProducerAck	2420
activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller (Mar- shaling code for Open Wire Format for ProducerAckMarshaller (p. 2424))	2424

activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller (Marshaling code for Open Wire Format for ProducerAckMarshaller (p. 2427))	2427
activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller (Marshaling code for Open Wire Format for ProducerAckMarshaller (p. 2431))	2431
activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller (Marshaling code for Open Wire Format for ProducerAckMarshaller (p. 2435))	2435
activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller (Marshaling code for Open Wire Format for ProducerAckMarshaller (p. 2439))	2439
activemq::cmsutil::ProducerCallback (Callback for sending a message to a CMS destination)	2443
activemq::cmsutil::CmsTemplate::ProducerExecutor	2444
activemq::commands::ProducerId	2446
activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller (Marshaling code for Open Wire Format for ProducerIdMarshaller (p. 2450))	2450
activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller (Marshaling code for Open Wire Format for ProducerIdMarshaller (p. 2454))	2454
activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller (Marshaling code for Open Wire Format for ProducerIdMarshaller (p. 2457))	2457
activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller (Marshaling code for Open Wire Format for ProducerIdMarshaller (p. 2461))	2461
activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller (Marshaling code for Open Wire Format for ProducerIdMarshaller (p. 2465))	2465
activemq::commands::ProducerInfo	2469
activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller (Marshaling code for Open Wire Format for ProducerInfoMarshaller (p. 2474))	2474
activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller (Marshaling code for Open Wire Format for ProducerInfoMarshaller (p. 2478))	2478
activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller (Marshaling code for Open Wire Format for ProducerInfoMarshaller (p. 2482))	2482
activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller (Marshaling code for Open Wire Format for ProducerInfoMarshaller (p. 2486))	2486
activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller (Marshaling code for Open Wire Format for ProducerInfoMarshaller (p. 2490))	2490
activemq::state::ProducerState	2494
decaf::util::Properties (Java-like properties class for mapping string names to string values)	2494
decaf::util::logging::PropertiesChangeListener (Defines the interface that classes can use to listen for change events on Properties (p. 2494))	2503
decaf::net::ProtocolException	2504
decaf::security::PublicKey (A public key)	2506

decaf::util::Queue< E > (A kind of collection provides advanced operations than other basic collections, such as insertion, extraction, and inspection)	2507
cms::Queue (An interface encapsulating a provider-specific queue name)	2510
cms::QueueBrowser (This class implements in interface for browsing the messages in a Queue (p. 2510) without removing them)	2511
decaf::util::Random (Random (p. 2513) Value Generator which is used to generate a stream of pseudorandom numbers)	2513
activemq::transport::inactivity::ReadChecker (Runnable class that is used by the {)	2517
decaf::io::Reader	2518
decaf::nio::ReadOnlyBufferException	2520
decaf::util::concurrent::locks::ReadWriteLock (A ReadWriteLock (p. 2522) maintains a pair of associated locks, one for read-only operations and one for writing)	2522
activemq::cmsutil::CmsTemplate::ReceiveExecutor	2524
decaf::util::concurrent::locks::ReentrantLock (A reentrant mutual exclusion Lock (p. 1898) with extended capabilities)	2526
decaf::util::concurrent::RejectedExecutionException	2533
decaf::util::concurrent::RejectedExecutionHandler (A handler for tasks that cannot be executed by a ThreadPoolExecutor (p. ??))	2536
activemq::commands::RemoveInfo	2536
activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller (Marshaling code for Open Wire Format for RemoveInfoMarshaller (p. 2540))	2540
activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller (Marshaling code for Open Wire Format for RemoveInfoMarshaller (p. 2544))	2544
activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller (Marshaling code for Open Wire Format for RemoveInfoMarshaller (p. 2548))	2548
activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller (Marshaling code for Open Wire Format for RemoveInfoMarshaller (p. 2552))	2552
activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller (Marshaling code for Open Wire Format for RemoveInfoMarshaller (p. 2556))	2556
activemq::commands::RemoveSubscriptionInfo	2560
activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller (Marshaling code for Open Wire Format for RemoveSubscriptionInfoMarshaller (p. 2564))	2564
activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller (Marshaling code for Open Wire Format for RemoveSubscriptionInfoMarshaller (p. 2568))	2568
activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller (Marshaling code for Open Wire Format for RemoveSubscriptionInfoMarshaller (p. 2572))	2572
activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller (Marshaling code for Open Wire Format for RemoveSubscriptionInfoMarshaller (p. 2576))	2576
activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller (Marshaling code for Open Wire Format for RemoveSubscriptionInfoMarshaller (p. 2580))	2580
activemq::commands::ReplayCommand	2584

activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller (Marshaling code for Open Wire Format for ReplayCommandMarshaller (p. 2587))	2587
activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller (Marshaling code for Open Wire Format for ReplayCommandMarshaller (p. 2591))	2591
activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller (Marshaling code for Open Wire Format for ReplayCommandMarshaller (p. 2595))	2595
activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller (Marshaling code for Open Wire Format for ReplayCommandMarshaller (p. 2599))	2599
activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller (Marshaling code for Open Wire Format for ReplayCommandMarshaller (p. 2603))	2603
activemq::cmsutil::CmsTemplate::ResolveProducerExecutor	2607
activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor	2608
activemq::cmsutil::ResourceLifecycleManager (Manages the lifecycle of a set of CMS resources)	2608
activemq::commands::Response	2611
activemq::transport::mock::ResponseBuilder (Interface for all Protocols to imple- ment that defines the behavior of the Broker in response to messages of that protocol)	2614
activemq::transport::correlator::ResponseCorrelator (This type of transport filter is responsible for correlating asynchronous responses with requests)	2616
activemq::wireformat::openwire::marshal::v2::ResponseMarshaller (Marshaling code for Open Wire Format for ResponseMarshaller (p. 2620))	2620
activemq::wireformat::openwire::marshal::v3::ResponseMarshaller (Marshaling code for Open Wire Format for ResponseMarshaller (p. 2624))	2624
activemq::wireformat::openwire::marshal::v5::ResponseMarshaller (Marshaling code for Open Wire Format for ResponseMarshaller (p. 2628))	2628
activemq::wireformat::openwire::marshal::v1::ResponseMarshaller (Marshaling code for Open Wire Format for ResponseMarshaller (p. 2633))	2633
activemq::wireformat::openwire::marshal::v4::ResponseMarshaller (Marshaling code for Open Wire Format for ResponseMarshaller (p. 2637))	2637
decaf::lang::Runnable (Interface for a runnable object - defines a task that can be run by a thread)	2642
decaf::lang::Runtime	2643
decaf::lang::exceptions::RuntimeException	2644
decaf::security_provider::SecurityProvider	2647
decaf::security_provider::SecurityProviderMap (Lookup Map for Connector Fac- tories)	2648
decaf::security_provider::SecurityProviderRegistrar (Registers the passed in provider into the provider map, this class can manage the lifetime of the regis- tered provider (default behaviour))	2649
decaf::util::concurrent::Semaphore (A counting semaphore)	2651
activemq::cmsutil::CmsTemplate::SendExecutor	2660
decaf::net::ServerSocket (A server socket class (for testing purposes))	2661
cms::Session (A Session (p. 2663) object is a single-threaded context for producing and consuming messages)	2663
activemq::cmsutil::SessionCallback (Callback for executing any number of opera- tions on a provided CMS Session)	2676
activemq::commands::SessionId	2677

activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller (Marshaling code for Open Wire Format for SessionIdMarshaller (p. 2681))	2681
activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller (Marshaling code for Open Wire Format for SessionIdMarshaller (p. 2685))	2685
activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller (Marshaling code for Open Wire Format for SessionIdMarshaller (p. 2689))	2689
activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller (Marshaling code for Open Wire Format for SessionIdMarshaller (p. 2693))	2693
activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller (Marshaling code for Open Wire Format for SessionIdMarshaller (p. 2697))	2697
activemq::commands::SessionInfo	2701
activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller (Marshaling code for Open Wire Format for SessionInfoMarshaller (p. 2705))	2705
activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller (Marshaling code for Open Wire Format for SessionInfoMarshaller (p. 2709))	2709
activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller (Marshaling code for Open Wire Format for SessionInfoMarshaller (p. 2713))	2713
activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller (Marshaling code for Open Wire Format for SessionInfoMarshaller (p. 2717))	2717
activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller (Marshaling code for Open Wire Format for SessionInfoMarshaller (p. 2721))	2721
activemq::cmsutil::SessionPool (A pool of CMS sessions from the same connection and with the same acknowledge mode)	2725
activemq::state::SessionState	2726
decaf::util::Set< E > (A collection that contains no duplicate elements)	2729
decaf::lang::Short	2729
decaf::internal::nio::ShortArrayBuffer	2738
decaf::nio::ShortBuffer (This class defines four categories of operations upon short buffers:)	2745
activemq::commands::ShutdownInfo	2757
activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller (Marshaling code for Open Wire Format for ShutdownInfoMarshaller (p. 2760))	2760
activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller (Marshaling code for Open Wire Format for ShutdownInfoMarshaller (p. 2763))	2763
activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller (Marshaling code for Open Wire Format for ShutdownInfoMarshaller (p. 2767))	2767
activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller (Marshaling code for Open Wire Format for ShutdownInfoMarshaller (p. 2771))	2771
activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller (Marshaling code for Open Wire Format for ShutdownInfoMarshaller (p. 2775))	2775
decaf::security::SignatureException	2779
decaf::util::logging::SimpleFormatter (Print a brief summary of the LogRecord (p. 1928) in a human readable format)	2782
decaf::util::logging::SimpleLogger	2784
decaf::net::Socket	2786
decaf::net::SocketError (Static utility class to simplify handling of error codes for socket operations)	2792
decaf::net::SocketException (Exception for errors when manipulating sockets)	2793

decaf::net::SocketFactory (Socket (p. 2786) Factory implementation for use in Creating Sockets)	2795
decaf::net::SocketInputStream (Input stream for performing reads on a socket) . .	2796
decaf::net::SocketOutputStream (Output stream for performing write operations on a socket)	2803
decaf::net::SocketTimeoutException	2809
activemq::commands::BrokerError::StackTraceElement	2811
decaf::internal::io::StandardErrorOutputStream (Wrapper Around the Standard error Output facility on the current platform)	2812
decaf::internal::io::StandardInputStream	2818
decaf::internal::io::StandardOutputStream	2825
cms::Startable (Interface for a class that implements the start method)	2831
decaf::lang::STATIC_CAST_TOKEN	2831
activemq::core::ActiveMQConstants::StaticInitializer	2832
decaf::util::StlList< E > (List (p. 1865) class that wraps the STL list object to provide a simpler interface and additional methods not provided by the STL type) . .	2833
decaf::util::StlMap< K, V, COMPARATOR > (Map (p. 1970) template that wraps around a std::map to provide a more user-friendly interface and to provide common functions that do not exist in std::map)	2845
decaf::util::StlQueue< T > (The Queue (p. 2507) class accepts messages with an psuh(m) command where m is the message to be queued)	2858
decaf::util::StlSet< E > (Set (p. 2729) template that wraps around a std::set to provide a more user-friendly interface and to provide common functions that do not exist in std::set)	2865
activemq::wireformat::stomp::StompCommandConstants	2871
activemq::wireformat::stomp::StompFrame (A Stomp-level message frame that encloses all messages to and from the broker)	2874
activemq::wireformat::stomp::StompHelper (Utility Methods used when marshaling to and from StompFrame's)	2878
activemq::wireformat::stomp::StompWireFormat	2883
activemq::wireformat::stomp::StompWireFormatFactory (Factory used to create the Stomp Wire Format instance)	2886
cms::Stoppable (Interface for a class that implements the stop method)	2887
decaf::util::logging::StreamHandler	2888
cms::StreamMessage (Interface for a StreamMessage (p. 2892))	2892
decaf::util::StringTokenizer	2904
activemq::commands::SubscriptionInfo	2907
activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller (Marshaling code for Open Wire Format for SubscriptionInfoMarshaller (p. 2911))	2911
activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller (Marshaling code for Open Wire Format for SubscriptionInfoMarshaller (p. 2915))	2915
activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller (Marshaling code for Open Wire Format for SubscriptionInfoMarshaller (p. 2919))	2919
activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller (Marshaling code for Open Wire Format for SubscriptionInfoMarshaller (p. 2923))	2923
activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller (Marshaling code for Open Wire Format for SubscriptionInfoMarshaller (p. 2926))	2926
decaf::util::concurrent::Synchronizable (The interface for all synchronizable objects (that is, objects that can be locked and unlocked))	2930

decaf::internal::util::concurrent::SynchronizableImpl (A convenience class used by some Decaf classes to implement the Synchronizable interface when there is no issues related to multiple inheritance)	2941
activemq::core::Synchronization (Transacted Object Synchronization (p. 2945), used to sync the events of a Transaction with the items in the Transaction)	2945
decaf::util::concurrent::SynchronousQueue< E > (A BlockingQueue blocking queue} in which each insert operation must wait for a corresponding remove operation by another thread, and vice versa)	2946
decaf::lang::System	2953
activemq::threads::Task (Represents a unit of work that requires one or more iterations to complete)	2956
decaf::util::concurrent::TaskListener	2957
activemq::threads::TaskRunner	2958
decaf::net::TcpSocket (Platform-independent implementation of the socket interface)	2959
activemq::transport::tcp::TcpTransport (Implements a TCP/IP based transport filter, this transport is meant to wrap an instance of an IOTransport (p. 1709))	2967
activemq::transport::tcp::TcpTransportFactory (Factory Responsible for creating the TcpTransport (p. 2967))	2969
cms::TemporaryQueue (Defines a Temporary Queue (p. 2510) based Destination (p. 1387))	2971
cms::TemporaryTopic (Defines a Temporary Topic (p. 3014) based Destination (p. 1387))	2973
cms::TextMessage (Interface for a text message)	2974
decaf::util::concurrent::ThreadFactory (Public interface ThreadFactory (p. 2976))	2976
decaf::lang::ThreadGroup	2977
decaf::util::concurrent::ThreadPool (Defines a Thread Pool object that implements the functionality of pooling threads to perform user tasks)	2977
decaf::lang::Throwable (This class represents an error that has occurred)	2983
decaf::util::concurrent::TimeoutException	2987
decaf::util::Timer (A facility for threads to schedule tasks for future execution in a background thread)	2989
decaf::util::TimerTask (A Base class for a task object that can be scheduled for one-time or repeated execution by a Timer (p. 2989))	3000
decaf::internal::util::TimerTaskHeap (A Binary Heap implemented specifically for the Timer class in Decaf Util)	3002
decaf::util::concurrent::TimeUnit (A TimeUnit (p. 3005) represents time durations at a given unit of granularity and provides utility methods to convert across units, and to perform timing and delay operations in these units)	3005
cms::Topic (An interface encapsulating a provider-specific topic name)	3014
activemq::state::Tracked	3015
activemq::commands::TransactionId	3015
activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller (Marshaling code for Open Wire Format for TransactionIdMarshaller (p. 3018))	3018
activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller (Marshaling code for Open Wire Format for TransactionIdMarshaller (p. 3022))	3022
activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller (Marshaling code for Open Wire Format for TransactionIdMarshaller (p. 3026))	3026

activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller (Marshaling code for Open Wire Format for TransactionIdMarshaller (p. 3029))	3029
activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller (Marshaling code for Open Wire Format for TransactionIdMarshaller (p. 3033))	3033
activemq::commands::TransactionInfo	3037
activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller (Marshaling code for Open Wire Format for TransactionInfoMarshaller (p. 3041))	3041
activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller (Marshaling code for Open Wire Format for TransactionInfoMarshaller (p. 3044))	3044
activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller (Marshaling code for Open Wire Format for TransactionInfoMarshaller (p. 3048))	3048
activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller (Marshaling code for Open Wire Format for TransactionInfoMarshaller (p. 3052))	3052
activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller (Marshaling code for Open Wire Format for TransactionInfoMarshaller (p. 3056))	3056
activemq::state::TransactionState	3060
decaf::internal::util::concurrent::Transferer< E > (Shared internal API for dual stacks and queues)	3062
decaf::internal::util::concurrent::TransferQueue< E > (This extends Scherer-Scott dual queue algorithm, differing, among other ways, by using modes within nodes rather than marked pointers)	3062
decaf::internal::util::concurrent::TransferStack< E >	3064
activemq::transport::Transport (Interface for a transport layer for command objects)	3066
activemq::transport::TransportFactory (Defines the interface for Factories that create Transports or TransportFilters)	3071
activemq::transport::TransportFilter (A filter on the transport layer)	3073
activemq::transport::TransportListener (A listener of asynchronous exceptions from a command transport object)	3080
activemq::transport::TransportRegistry (Registry of all Transport (p. 3066) Factories that are available to the client at runtime)	3082
decaf::net::UnknownHostException	3085
decaf::net::UnknownServiceException	3087
decaf::io::UnsupportedEncodingException (Thrown when the the Character Encoding is not supported)	3090
cms::UnsupportedOperationException (This exception must be thrown when a CMS client attempts use a CMS method that is not implemented or not supported by the CMS Provider in use)	3093
decaf::lang::exceptions::UnsupportedOperationException	3094
decaf::net::URI (This class represents an instance of a URI (p. 3096) as defined by RFC 2396)	3096
decaf::internal::net::URIEncoderDecoder	3107
decaf::internal::net::URIHelper (Helper class used by the URI classes in encoding and decoding of URI's)	3110
activemq::transport::failover::URIPool	3117
activemq::util::URISupport	3119
decaf::net::URISyntaxException	3122

decaf::internal::net::URIType (Basic type object that holds data that composes a given URI)	3126
decaf::net::URL (Class URL (p.3133) represents a Uniform Resource Locator, a pointer to a "resource" on the World Wide Web)	3133
decaf::net::URLDecoder	3135
decaf::net::URLEncoder	3136
activemq::util::Usage	3137
decaf::io::UTFDataFormatException (Thrown from classes that attempt to read or write a UTF-8 encoded string and an encoding error is encountered)	3139
decaf::util::UUID (A class that represents an immutable universally unique identifier (UUID (p. 3141)))	3141
activemq::wireformat::WireFormat (Provides a mechanism to marshal commands into and out of packets or into and out of streams, Channels and Datagrams) .	3148
activemq::wireformat::WireFormatFactory (The WireFormatFactory (p.3151) is the interface that all WireFormatFactory (p.3151) classes must extend) .	3151
activemq::commands::WireFormatInfo	3152
activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller (Marshaling code for Open Wire Format for WireFormatInfoMarshaller (p.3163))	3163
activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller (Marshaling code for Open Wire Format for WireFormatInfoMarshaller (p.3166))	3166
activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller (Marshaling code for Open Wire Format for WireFormatInfoMarshaller (p.3170))	3170
activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller (Marshaling code for Open Wire Format for WireFormatInfoMarshaller (p.3174))	3174
activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller (Marshaling code for Open Wire Format for WireFormatInfoMarshaller (p.3178))	3178
activemq::wireformat::WireFormatNegotiator (Defines a WireFormatNegotiator (p.3182) which allows a WireFormat (p.3148) to)	3182
activemq::wireformat::WireFormatRegistry (Registry of all WireFormat (p.3148) Factories that are available to the client at runtime)	3183
activemq::transport::inactivity::WriteChecker (Runnable class used by the {) .	3186
decaf::io::Writer	3187
decaf::security::auth::x500::X500Principal	3188
decaf::security::cert::X509Certificate (Base interface for all identity certificates) .	3189
activemq::commands::XATransactionId	3192
activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller (Marshaling code for Open Wire Format for XATransactionIdMarshaller (p.3197))	3197
activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller (Marshaling code for Open Wire Format for XATransactionIdMarshaller (p.3201))	3201
activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller (Marshaling code for Open Wire Format for XATransactionIdMarshaller (p.3205))	3205
activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller (Marshaling code for Open Wire Format for XATransactionIdMarshaller (p.3209))	3209

activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller	
(Marshaling code for Open Wire Format for XATransactionIdMarshaller	
(p. 3213)	3213

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

src/main/activemq/cmsutil/	CachedConsumer.h	3219
src/main/activemq/cmsutil/	CachedProducer.h	3219
src/main/activemq/cmsutil/	CmsAccessor.h	3220
src/main/activemq/cmsutil/	CmsDestinationAccessor.h	3220
src/main/activemq/cmsutil/	CmsTemplate.h	3221
src/main/activemq/cmsutil/	DestinationResolver.h	3221
src/main/activemq/cmsutil/	DynamicDestinationResolver.h	3222
src/main/activemq/cmsutil/	MessageCreator.h	3222
src/main/activemq/cmsutil/	PooledSession.h	3223
src/main/activemq/cmsutil/	ProducerCallback.h	3223
src/main/activemq/cmsutil/	ResourceLifecycleManager.h	3224
src/main/activemq/cmsutil/	SessionCallback.h	3225
src/main/activemq/cmsutil/	SessionPool.h	3225
src/main/activemq/commands/	ActiveMQBlobMessage.h	3226
src/main/activemq/commands/	ActiveMQBytesMessage.h	3226
src/main/activemq/commands/	ActiveMQDestination.h	3227
src/main/activemq/commands/	ActiveMQMapMessage.h	3227
src/main/activemq/commands/	ActiveMQMessage.h	3228
src/main/activemq/commands/	ActiveMQMessageTemplate.h	3228
src/main/activemq/commands/	ActiveMQObjectMessage.h	3229
src/main/activemq/commands/	ActiveMQQueue.h	3229
src/main/activemq/commands/	ActiveMQStreamMessage.h	3230
src/main/activemq/commands/	ActiveMQTempDestination.h	3231
src/main/activemq/commands/	ActiveMQTempQueue.h	3231
src/main/activemq/commands/	ActiveMQTempTopic.h	3232
src/main/activemq/commands/	ActiveMQTextMessage.h	3232
src/main/activemq/commands/	ActiveMQTopic.h	3233
src/main/activemq/commands/	BaseCommand.h	3233
src/main/activemq/commands/	BaseDataStructure.h	3234
src/main/activemq/commands/	BooleanExpression.h	3234
src/main/activemq/commands/	BrokerError.h	3235
src/main/activemq/commands/	BrokerId.h	3235
src/main/activemq/commands/	BrokerInfo.h	3236

src/main/activemq/commands/	Command.h	3236
src/main/activemq/commands/	ConnectionControl.h	3237
src/main/activemq/commands/	ConnectionError.h	3237
src/main/activemq/commands/	ConnectionId.h	3238
src/main/activemq/commands/	ConnectionInfo.h	3238
src/main/activemq/commands/	ConsumerControl.h	3239
src/main/activemq/commands/	ConsumerId.h	3239
src/main/activemq/commands/	ConsumerInfo.h	3240
src/main/activemq/commands/	ControlCommand.h	3241
src/main/activemq/commands/	DataArrayResponse.h	3241
src/main/activemq/commands/	DataResponse.h	3242
src/main/activemq/commands/	DataStructure.h	3242
src/main/activemq/commands/	DestinationInfo.h	3242
src/main/activemq/commands/	DiscoveryEvent.h	3243
src/main/activemq/commands/	ExceptionResponse.h	3244
src/main/activemq/commands/	FlushCommand.h	3244
src/main/activemq/commands/	IntegerResponse.h	3245
src/main/activemq/commands/	JournalQueueAck.h	3245
src/main/activemq/commands/	JournalTopicAck.h	3246
src/main/activemq/commands/	JournalTrace.h	3246
src/main/activemq/commands/	JournalTransaction.h	3247
src/main/activemq/commands/	KeepAliveInfo.h	3247
src/main/activemq/commands/	LastPartialCommand.h	3248
src/main/activemq/commands/	LocalTransactionId.h	3248
src/main/activemq/commands/	Message.h	3249
src/main/activemq/commands/	MessageAck.h	3250
src/main/activemq/commands/	MessageDispatch.h	3251
src/main/activemq/commands/	MessageDispatchNotification.h	3251
src/main/activemq/commands/	MessageId.h	3252
src/main/activemq/commands/	MessagePull.h	3252
src/main/activemq/commands/	NetworkBridgeFilter.h	3253
src/main/activemq/commands/	PartialCommand.h	3253
src/main/activemq/commands/	ProducerAck.h	3254
src/main/activemq/commands/	ProducerId.h	3254
src/main/activemq/commands/	ProducerInfo.h	3255
src/main/activemq/commands/	RemoveInfo.h	3255
src/main/activemq/commands/	RemoveSubscriptionInfo.h	3256
src/main/activemq/commands/	ReplayCommand.h	3256
src/main/activemq/commands/	Response.h	3257
src/main/activemq/commands/	SessionId.h	3257
src/main/activemq/commands/	SessionInfo.h	3258
src/main/activemq/commands/	ShutdownInfo.h	3259
src/main/activemq/commands/	SubscriptionInfo.h	3259
src/main/activemq/commands/	TransactionId.h	3260
src/main/activemq/commands/	TransactionInfo.h	3260
src/main/activemq/commands/	WireFormatInfo.h	3261
src/main/activemq/commands/	XATransactionId.h	3261
src/main/activemq/core/	ActiveMQAckHandler.h	3262
src/main/activemq/core/	ActiveMQConnection.h	3262
src/main/activemq/core/	ActiveMQConnectionFactory.h	3263
src/main/activemq/core/	ActiveMQConnectionMetaData.h	3264
src/main/activemq/core/	ActiveMQConnectionSupport.h	3264
src/main/activemq/core/	ActiveMQConstants.h	3265
src/main/activemq/core/	ActiveMQConsumer.h	3265

src/main/activemq/core/ActiveMQProducer.h	3266
src/main/activemq/core/ActiveMQSession.h	3267
src/main/activemq/core/ActiveMQSessionExecutor.h	3268
src/main/activemq/core/ActiveMQTransactionContext.h	3268
src/main/activemq/core/DispatchData.h	3269
src/main/activemq/core/Dispatcher.h	3269
src/main/activemq/core/MessageDispatchChannel.h	3270
src/main/activemq/core/Synchronization.h	3270
src/main/activemq/exceptions/ActiveMQException.h	3271
src/main/activemq/exceptions/BrokerException.h	3271
src/main/activemq/exceptions/ExceptionDefines.h	3272
src/main/activemq/io/LoggingInputStream.h	3276
src/main/activemq/io/LoggingOutputStream.h	3277
src/main/activemq/library/ActiveMQCPP.h	3277
src/main/activemq/state/CommandVisitor.h	3278
src/main/activemq/state/CommandVisitorAdapter.h	3278
src/main/activemq/state/ConnectionState.h	3279
src/main/activemq/state/ConnectionStateTracker.h	3280
src/main/activemq/state/ConsumerState.h	3281
src/main/activemq/state/ProducerState.h	3281
src/main/activemq/state/SessionState.h	3282
src/main/activemq/state/Tracked.h	3282
src/main/activemq/state/TransactionState.h	3283
src/main/activemq/threads/CompositeTask.h	3284
src/main/activemq/threads/CompositeTaskRunner.h	3284
src/main/activemq/threads/DedicatedTaskRunner.h	3285
src/main/activemq/threads/Task.h	3285
src/main/activemq/threads/TaskRunner.h	3286
src/main/activemq/transport/AbstractTransportFactory.h	3286
src/main/activemq/transport/CompositeTransport.h	3287
src/main/activemq/transport/DefaultTransportListener.h	3288
src/main/activemq/transport/IOTransport.h	3295
src/main/activemq/transport/Transport.h	3300
src/main/activemq/transport/TransportFactory.h	3300
src/main/activemq/transport/TransportFilter.h	3301
src/main/activemq/transport/TransportListener.h	3302
src/main/activemq/transport/TransportRegistry.h	3302
src/main/activemq/transport/correlator/FutureResponse.h	3287
src/main/activemq/transport/correlator/ResponseCorrelator.h	3288
src/main/activemq/transport/failover/BackupTransport.h	3289
src/main/activemq/transport/failover/BackupTransportPool.h	3289
src/main/activemq/transport/failover/CloseTransportsTask.h	3290
src/main/activemq/transport/failover/FailoverTransport.h	3290
src/main/activemq/transport/failover/FailoverTransportFactory.h	3291
src/main/activemq/transport/failover/FailoverTransportListener.h	3292
src/main/activemq/transport/failover/URIPool.h	3292
src/main/activemq/transport/inactivity/InactivityMonitor.h	3293
src/main/activemq/transport/inactivity/ReadChecker.h	3294
src/main/activemq/transport/inactivity/WriteChecker.h	3294
src/main/activemq/transport/logging/LoggingTransport.h	3295
src/main/activemq/transport/mock/InternalCommandListener.h	3296
src/main/activemq/transport/mock/MockTransport.h	3297
src/main/activemq/transport/mock/MockTransportFactory.h	3297
src/main/activemq/transport/mock/ResponseBuilder.h	3298

src/main/activemq/transport/tcp/ TcpTransport.h	3298
src/main/activemq/transport/tcp/ TcpTransportFactory.h	3299
src/main/activemq/util/ ActiveMQProperties.h	3303
src/main/activemq/util/ CMSExceptionSupport.h	3303
src/main/activemq/util/ CompositeData.h	3305
src/main/activemq/util/ Config.h	3305
src/main/activemq/util/ LongSequenceGenerator.h	3306
src/main/activemq/util/ MemoryUsage.h	3307
src/main/activemq/util/ PrimitiveList.h	3307
src/main/activemq/util/ PrimitiveMap.h	3308
src/main/activemq/util/ PrimitiveValueConverter.h	3309
src/main/activemq/util/ PrimitiveValueNode.h	3309
src/main/activemq/util/ URISupport.h	3310
src/main/activemq/util/ Usage.h	3310
src/main/activemq/wireformat/ MarshalAware.h	3310
src/main/activemq/wireformat/ WireFormat.h	3527
src/main/activemq/wireformat/ WireFormatFactory.h	3527
src/main/activemq/wireformat/ WireFormatNegotiator.h	3528
src/main/activemq/wireformat/ WireFormatRegistry.h	3528
src/main/activemq/wireformat/openwire/ OpenWireFormat.h	3519
src/main/activemq/wireformat/openwire/ OpenWireFormatFactory.h	3520
src/main/activemq/wireformat/openwire/ OpenWireFormatNegotiator.h	3520
src/main/activemq/wireformat/openwire/ OpenWireResponseBuilder.h	3521
src/main/activemq/wireformat/openwire/marshal/ BaseDataStreamMarshaller.h	3311
src/main/activemq/wireformat/openwire/marshal/ DataStreamMarshaller.h	3312
src/main/activemq/wireformat/openwire/marshal/ PrimitiveTypesMarshaller.h	3312
src/main/activemq/wireformat/openwire/marshal/v1/ ActiveMQBlobMessageMarshaller.h	3313
src/main/activemq/wireformat/openwire/marshal/v1/ ActiveMQBytesMessageMarshaller.h	3316
src/main/activemq/wireformat/openwire/marshal/v1/ ActiveMQDestinationMarshaller.h	3320
src/main/activemq/wireformat/openwire/marshal/v1/ ActiveMQMapMessageMarshaller.h	3323
src/main/activemq/wireformat/openwire/marshal/v1/ ActiveMQMessageMarshaller.h	3326
src/main/activemq/wireformat/openwire/marshal/v1/ ActiveMQObjectMessageMarshaller.h	3330
src/main/activemq/wireformat/openwire/marshal/v1/ ActiveMQQueueMarshaller.h	3333
src/main/activemq/wireformat/openwire/marshal/v1/ ActiveMQStreamMessageMarshaller.h	3336
src/main/activemq/wireformat/openwire/marshal/v1/ ActiveMQTempDestinationMarshaller.h	3340
src/main/activemq/wireformat/openwire/marshal/v1/ ActiveMQTempQueueMarshaller.h	3343
src/main/activemq/wireformat/openwire/marshal/v1/ ActiveMQTempTopicMarshaller.h	3346
src/main/activemq/wireformat/openwire/marshal/v1/ ActiveMQTextMessageMarshaller.h	3350
src/main/activemq/wireformat/openwire/marshal/v1/ ActiveMQTopicMarshaller.h	3353
src/main/activemq/wireformat/openwire/marshal/v1/ BaseCommandMarshaller.h	3356
src/main/activemq/wireformat/openwire/marshal/v1/ BrokerIdMarshaller.h	3360
src/main/activemq/wireformat/openwire/marshal/v1/ BrokerInfoMarshaller.h	3363

src/main/activemq/wireformat/openwire/marshal/v1/	ConnectionControlMarshaller.h	
3366		
src/main/activemq/wireformat/openwire/marshal/v1/	ConnectionErrorMarshaller.h	
3370		
src/main/activemq/wireformat/openwire/marshal/v1/	ConnectionIdMarshaller.h	3373
src/main/activemq/wireformat/openwire/marshal/v1/	ConnectionInfoMarshaller.h	3376
src/main/activemq/wireformat/openwire/marshal/v1/	ConsumerControlMarshaller.h	
3380		
src/main/activemq/wireformat/openwire/marshal/v1/	ConsumerIdMarshaller.h	3383
src/main/activemq/wireformat/openwire/marshal/v1/	ConsumerInfoMarshaller.h	3386
src/main/activemq/wireformat/openwire/marshal/v1/	ControlCommandMarshaller.h	
3390		
src/main/activemq/wireformat/openwire/marshal/v1/	DataArrayResponseMarshaller.h	
3393		
src/main/activemq/wireformat/openwire/marshal/v1/	DataResponseMarshaller.h	3396
src/main/activemq/wireformat/openwire/marshal/v1/	DestinationInfoMarshaller.h	3400
src/main/activemq/wireformat/openwire/marshal/v1/	DiscoveryEventMarshaller.h	3403
src/main/activemq/wireformat/openwire/marshal/v1/	ExceptionResponseMarshaller.h	
3406		
src/main/activemq/wireformat/openwire/marshal/v1/	FlushCommandMarshaller.h	3410
src/main/activemq/wireformat/openwire/marshal/v1/	IntegerResponseMarshaller.h	
3413		
src/main/activemq/wireformat/openwire/marshal/v1/	JournalQueueAckMarshaller.h	
3416		
src/main/activemq/wireformat/openwire/marshal/v1/	JournalTopicAckMarshaller.h	
3420		
src/main/activemq/wireformat/openwire/marshal/v1/	JournalTraceMarshaller.h	3423
src/main/activemq/wireformat/openwire/marshal/v1/	JournalTransactionMarshaller.h	
3426		
src/main/activemq/wireformat/openwire/marshal/v1/	KeepAliveInfoMarshaller.h	3430
src/main/activemq/wireformat/openwire/marshal/v1/	LastPartialCommandMarshaller.h	
3433		
src/main/activemq/wireformat/openwire/marshal/v1/	LocalTransactionIdMarshaller.h	
3436		
src/main/activemq/wireformat/openwire/marshal/v1/	MarshallerFactory.h	3440
src/main/activemq/wireformat/openwire/marshal/v1/	MessageAckMarshaller.h	3442
src/main/activemq/wireformat/openwire/marshal/v1/	MessageDispatchMarshaller.h	
3445		
src/main/activemq/wireformat/openwire/marshal/v1/	MessageDispatchNotificationMarshaller.h	
3449		
src/main/activemq/wireformat/openwire/marshal/v1/	MessageIdMarshaller.h	3452
src/main/activemq/wireformat/openwire/marshal/v1/	MessageMarshaller.h	3456
src/main/activemq/wireformat/openwire/marshal/v1/	MessagePullMarshaller.h	3459
src/main/activemq/wireformat/openwire/marshal/v1/	NetworkBridgeFilterMarshaller.h	
3462		
src/main/activemq/wireformat/openwire/marshal/v1/	PartialCommandMarshaller.h	
3466		
src/main/activemq/wireformat/openwire/marshal/v1/	ProducerAckMarshaller.h	3469
src/main/activemq/wireformat/openwire/marshal/v1/	ProducerIdMarshaller.h	3472
src/main/activemq/wireformat/openwire/marshal/v1/	ProducerInfoMarshaller.h	3476
src/main/activemq/wireformat/openwire/marshal/v1/	RemoveInfoMarshaller.h	3479
src/main/activemq/wireformat/openwire/marshal/v1/	RemoveSubscriptionInfoMarshaller.h	
3482		

src/main/activemq/wireformat/openwire/marshal/v1/ ReplayCommandMarshaller.h	
3486	
src/main/activemq/wireformat/openwire/marshal/v1/ ResponseMarshaller.h	3489
src/main/activemq/wireformat/openwire/marshal/v1/ SessionIdMarshaller.h	3492
src/main/activemq/wireformat/openwire/marshal/v1/ SessionInfoMarshaller.h . . .	3496
src/main/activemq/wireformat/openwire/marshal/v1/ ShutdownInfoMarshaller.h .	3499
src/main/activemq/wireformat/openwire/marshal/v1/ SubscriptionInfoMarshaller.h	
3502	
src/main/activemq/wireformat/openwire/marshal/v1/ TransactionIdMarshaller.h .	3506
src/main/activemq/wireformat/openwire/marshal/v1/ TransactionInfoMarshaller.h	3509
src/main/activemq/wireformat/openwire/marshal/v1/ WireFormatInfoMarshaller.h	3512
src/main/activemq/wireformat/openwire/marshal/v1/ XATransactionIdMarshaller.h	
3516	
src/main/activemq/wireformat/openwire/marshal/v2/ ActiveMQBlobMessageMarshaller.h	
3314	
src/main/activemq/wireformat/openwire/marshal/v2/ ActiveMQBytesMessageMarshaller.h	
3317	
src/main/activemq/wireformat/openwire/marshal/v2/ ActiveMQDestinationMarshaller.h	
3320	
src/main/activemq/wireformat/openwire/marshal/v2/ ActiveMQMapMessageMarshaller.h	
3324	
src/main/activemq/wireformat/openwire/marshal/v2/ ActiveMQMessageMarshaller.h	
3327	
src/main/activemq/wireformat/openwire/marshal/v2/ ActiveMQObjectMessageMarshaller.h	
3330	
src/main/activemq/wireformat/openwire/marshal/v2/ ActiveMQQueueMarshaller.h	
3334	
src/main/activemq/wireformat/openwire/marshal/v2/ ActiveMQStreamMessageMarshaller.h	
3337	
src/main/activemq/wireformat/openwire/marshal/v2/ ActiveMQTempDestinationMarshaller.h	
3340	
src/main/activemq/wireformat/openwire/marshal/v2/ ActiveMQTempQueueMarshaller.h	
3344	
src/main/activemq/wireformat/openwire/marshal/v2/ ActiveMQTempTopicMarshaller.h	
3347	
src/main/activemq/wireformat/openwire/marshal/v2/ ActiveMQTextMessageMarshaller.h	
3350	
src/main/activemq/wireformat/openwire/marshal/v2/ ActiveMQTopicMarshaller.h	3354
src/main/activemq/wireformat/openwire/marshal/v2/ BaseCommandMarshaller.h	3357
src/main/activemq/wireformat/openwire/marshal/v2/ BrokerIdMarshaller.h	3360
src/main/activemq/wireformat/openwire/marshal/v2/ BrokerInfoMarshaller.h . . .	3364
src/main/activemq/wireformat/openwire/marshal/v2/ ConnectionControlMarshaller.h	
3367	
src/main/activemq/wireformat/openwire/marshal/v2/ ConnectionErrorMarshaller.h	
3370	
src/main/activemq/wireformat/openwire/marshal/v2/ ConnectionIdMarshaller.h .	3374
src/main/activemq/wireformat/openwire/marshal/v2/ ConnectionInfoMarshaller.h	3377
src/main/activemq/wireformat/openwire/marshal/v2/ ConsumerControlMarshaller.h	
3380	
src/main/activemq/wireformat/openwire/marshal/v2/ ConsumerIdMarshaller.h . .	3384
src/main/activemq/wireformat/openwire/marshal/v2/ ConsumerInfoMarshaller.h .	3387
src/main/activemq/wireformat/openwire/marshal/v2/ ControlCommandMarshaller.h	
3390	

src/main/activemq/wireformat/openwire/marshall/v2/DataArrayResponseMarshaller.h	
3394	
src/main/activemq/wireformat/openwire/marshall/v2/DataResponseMarshaller.h	3397
src/main/activemq/wireformat/openwire/marshall/v2/DestinationInfoMarshaller.h	3400
src/main/activemq/wireformat/openwire/marshall/v2/DiscoveryEventMarshaller.h	3404
src/main/activemq/wireformat/openwire/marshall/v2/ExceptionResponseMarshaller.h	3407
src/main/activemq/wireformat/openwire/marshall/v2/FlushCommandMarshaller.h	3410
src/main/activemq/wireformat/openwire/marshall/v2/IntegerResponseMarshaller.h	3414
src/main/activemq/wireformat/openwire/marshall/v2/JournalQueueAckMarshaller.h	3417
src/main/activemq/wireformat/openwire/marshall/v2/JournalTopicAckMarshaller.h	3420
src/main/activemq/wireformat/openwire/marshall/v2/JournalTraceMarshaller.h	3424
src/main/activemq/wireformat/openwire/marshall/v2/JournalTransactionMarshaller.h	3427
src/main/activemq/wireformat/openwire/marshall/v2/KeepAliveInfoMarshaller.h	3430
src/main/activemq/wireformat/openwire/marshall/v2/LastPartialCommandMarshaller.h	3434
src/main/activemq/wireformat/openwire/marshall/v2/LocalTransactionIdMarshaller.h	3437
src/main/activemq/wireformat/openwire/marshall/v2/MarshallerFactory.h	3440
src/main/activemq/wireformat/openwire/marshall/v2/MessageAckMarshaller.h	3443
src/main/activemq/wireformat/openwire/marshall/v2/MessageDispatchMarshaller.h	3446
src/main/activemq/wireformat/openwire/marshall/v2/MessageDispatchNotificationMarshaller.h	3450
src/main/activemq/wireformat/openwire/marshall/v2/MessageIdMarshaller.h	3453
src/main/activemq/wireformat/openwire/marshall/v2/MessageMarshaller.h	3456
src/main/activemq/wireformat/openwire/marshall/v2/MessagePullMarshaller.h	3460
src/main/activemq/wireformat/openwire/marshall/v2/NetworkBridgeFilterMarshaller.h	3463
src/main/activemq/wireformat/openwire/marshall/v2/PartialCommandMarshaller.h	3466
src/main/activemq/wireformat/openwire/marshall/v2/ProducerAckMarshaller.h	3470
src/main/activemq/wireformat/openwire/marshall/v2/ProducerIdMarshaller.h	3473
src/main/activemq/wireformat/openwire/marshall/v2/ProducerInfoMarshaller.h	3476
src/main/activemq/wireformat/openwire/marshall/v2/RemoveInfoMarshaller.h	3480
src/main/activemq/wireformat/openwire/marshall/v2/RemoveSubscriptionInfoMarshaller.h	3483
src/main/activemq/wireformat/openwire/marshall/v2/ReplayCommandMarshaller.h	3486
src/main/activemq/wireformat/openwire/marshall/v2/ResponseMarshaller.h	3490
src/main/activemq/wireformat/openwire/marshall/v2/SessionIdMarshaller.h	3493
src/main/activemq/wireformat/openwire/marshall/v2/SessionInfoMarshaller.h	3496
src/main/activemq/wireformat/openwire/marshall/v2/ShutdownInfoMarshaller.h	3500
src/main/activemq/wireformat/openwire/marshall/v2/SubscriptionInfoMarshaller.h	3503
src/main/activemq/wireformat/openwire/marshall/v2/TransactionIdMarshaller.h	3506
src/main/activemq/wireformat/openwire/marshall/v2/TransactionInfoMarshaller.h	3510
src/main/activemq/wireformat/openwire/marshall/v2/WireFormatInfoMarshaller.h	3513
src/main/activemq/wireformat/openwire/marshall/v2/XATransactionIdMarshaller.h	3516

src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQBlobMessageMarshaller.h	
3314	
src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQBytesMessageMarshaller.h	
3318	
src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQDestinationMarshaller.h	
3321	
src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQMapMessageMarshaller.h	
3324	
src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQMessageMarshaller.h	
3328	
src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQObjectMessageMarshaller.h	
3331	
src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQQueueMarshaller.h	
3334	
src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQStreamMessageMarshaller.h	
3338	
src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTempDestinationMarshaller.h	
3341	
src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTempQueueMarshaller.h	
3344	
src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTempTopicMarshaller.h	
3348	
src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTextMessageMarshaller.h	
3351	
src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTopicMarshaller.h	3354
src/main/activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h	3358
src/main/activemq/wireformat/openwire/marshal/v3/BrokerIdMarshaller.h	3361
src/main/activemq/wireformat/openwire/marshal/v3/BrokerInfoMarshaller.h	3364
src/main/activemq/wireformat/openwire/marshal/v3/ConnectionControlMarshaller.h	
3368	
src/main/activemq/wireformat/openwire/marshal/v3/ConnectionErrorMarshaller.h	
3371	
src/main/activemq/wireformat/openwire/marshal/v3/ConnectionIdMarshaller.h	3374
src/main/activemq/wireformat/openwire/marshal/v3/ConnectionInfoMarshaller.h	3378
src/main/activemq/wireformat/openwire/marshal/v3/ConsumerControlMarshaller.h	
3381	
src/main/activemq/wireformat/openwire/marshal/v3/ConsumerIdMarshaller.h	3384
src/main/activemq/wireformat/openwire/marshal/v3/ConsumerInfoMarshaller.h	3388
src/main/activemq/wireformat/openwire/marshal/v3/ControlCommandMarshaller.h	
3391	
src/main/activemq/wireformat/openwire/marshal/v3/DataArrayResponseMarshaller.h	
3394	
src/main/activemq/wireformat/openwire/marshal/v3/DataResponseMarshaller.h	3398
src/main/activemq/wireformat/openwire/marshal/v3/DestinationInfoMarshaller.h	3401
src/main/activemq/wireformat/openwire/marshal/v3/DiscoveryEventMarshaller.h	3404
src/main/activemq/wireformat/openwire/marshal/v3/ExceptionResponseMarshaller.h	
3408	
src/main/activemq/wireformat/openwire/marshal/v3/FlushCommandMarshaller.h	3411
src/main/activemq/wireformat/openwire/marshal/v3/IntegerResponseMarshaller.h	
3414	
src/main/activemq/wireformat/openwire/marshal/v3/JournalQueueAckMarshaller.h	
3418	
src/main/activemq/wireformat/openwire/marshal/v3/JournalTopicAckMarshaller.h	
3421	

src/main/activemq/wireformat/openwire/marshal/v3/ JournalTraceMarshaller.h . . .	3424
src/main/activemq/wireformat/openwire/marshal/v3/ JournalTransactionMarshaller.h	3428
src/main/activemq/wireformat/openwire/marshal/v3/ KeepAliveInfoMarshaller.h . .	3431
src/main/activemq/wireformat/openwire/marshal/v3/ LastPartialCommandMarshaller.h	3434
src/main/activemq/wireformat/openwire/marshal/v3/ LocalTransactionIdMarshaller.h	3438
src/main/activemq/wireformat/openwire/marshal/v3/ MarshallerFactory.h	3441
src/main/activemq/wireformat/openwire/marshal/v3/ MessageAckMarshaller.h . .	3443
src/main/activemq/wireformat/openwire/marshal/v3/ MessageDispatchMarshaller.h	3447
src/main/activemq/wireformat/openwire/marshal/v3/ MessageDispatchNotificationMarshaller.h	3450
src/main/activemq/wireformat/openwire/marshal/v3/ MessageIdMarshaller.h . . .	3454
src/main/activemq/wireformat/openwire/marshal/v3/ MessageMarshaller.h	3457
src/main/activemq/wireformat/openwire/marshal/v3/ MessagePullMarshaller.h . .	3460
src/main/activemq/wireformat/openwire/marshal/v3/ NetworkBridgeFilterMarshaller.h	3464
src/main/activemq/wireformat/openwire/marshal/v3/ PartialCommandMarshaller.h	3467
src/main/activemq/wireformat/openwire/marshal/v3/ ProducerAckMarshaller.h . .	3470
src/main/activemq/wireformat/openwire/marshal/v3/ ProducerIdMarshaller.h . . .	3474
src/main/activemq/wireformat/openwire/marshal/v3/ ProducerInfoMarshaller.h . .	3477
src/main/activemq/wireformat/openwire/marshal/v3/ RemoveInfoMarshaller.h . .	3480
src/main/activemq/wireformat/openwire/marshal/v3/ RemoveSubscriptionInfoMarshaller.h	3484
src/main/activemq/wireformat/openwire/marshal/v3/ ReplayCommandMarshaller.h	3487
src/main/activemq/wireformat/openwire/marshal/v3/ ResponseMarshaller.h	3490
src/main/activemq/wireformat/openwire/marshal/v3/ SessionIdMarshaller.h	3494
src/main/activemq/wireformat/openwire/marshal/v3/ SessionInfoMarshaller.h . . .	3497
src/main/activemq/wireformat/openwire/marshal/v3/ ShutdownInfoMarshaller.h .	3500
src/main/activemq/wireformat/openwire/marshal/v3/ SubscriptionInfoMarshaller.h	3504
src/main/activemq/wireformat/openwire/marshal/v3/ TransactionIdMarshaller.h .	3507
src/main/activemq/wireformat/openwire/marshal/v3/ TransactionInfoMarshaller.h	3510
src/main/activemq/wireformat/openwire/marshal/v3/ WireFormatInfoMarshaller.h	3514
src/main/activemq/wireformat/openwire/marshal/v3/ XATransactionIdMarshaller.h	3517
src/main/activemq/wireformat/openwire/marshal/v4/ ActiveMQBlobMessageMarshaller.h	3315
src/main/activemq/wireformat/openwire/marshal/v4/ ActiveMQBytesMessageMarshaller.h	3318
src/main/activemq/wireformat/openwire/marshal/v4/ ActiveMQDestinationMarshaller.h	3322
src/main/activemq/wireformat/openwire/marshal/v4/ ActiveMQMapMessageMarshaller.h	3325
src/main/activemq/wireformat/openwire/marshal/v4/ ActiveMQMessageMarshaller.h	3328
src/main/activemq/wireformat/openwire/marshal/v4/ ActiveMQObjectMessageMarshaller.h	3332
src/main/activemq/wireformat/openwire/marshal/v4/ ActiveMQQueueMarshaller.h	3335

src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQStreamMessageMarshaller.h	
3338	
src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTempDestinationMarshaller.h	
3342	
src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTempQueueMarshaller.h	
3345	
src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTempTopicMarshaller.h	
3348	
src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTextMessageMarshaller.h	
3352	
src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTopicMarshaller.h	3355
src/main/activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h	3358
src/main/activemq/wireformat/openwire/marshal/v4/BrokerIdMarshaller.h	3362
src/main/activemq/wireformat/openwire/marshal/v4/BrokerInfoMarshaller.h	3365
src/main/activemq/wireformat/openwire/marshal/v4/ConnectionControlMarshaller.h	
3368	
src/main/activemq/wireformat/openwire/marshal/v4/ConnectionErrorMarshaller.h	
3372	
src/main/activemq/wireformat/openwire/marshal/v4/ConnectionIdMarshaller.h	3375
src/main/activemq/wireformat/openwire/marshal/v4/ConnectionInfoMarshaller.h	3378
src/main/activemq/wireformat/openwire/marshal/v4/ConsumerControlMarshaller.h	
3382	
src/main/activemq/wireformat/openwire/marshal/v4/ConsumerIdMarshaller.h	3385
src/main/activemq/wireformat/openwire/marshal/v4/ConsumerInfoMarshaller.h	3388
src/main/activemq/wireformat/openwire/marshal/v4/ControlCommandMarshaller.h	
3392	
src/main/activemq/wireformat/openwire/marshal/v4/DataArrayResponseMarshaller.h	
3395	
src/main/activemq/wireformat/openwire/marshal/v4/DataResponseMarshaller.h	3398
src/main/activemq/wireformat/openwire/marshal/v4/DestinationInfoMarshaller.h	3402
src/main/activemq/wireformat/openwire/marshal/v4/DiscoveryEventMarshaller.h	3405
src/main/activemq/wireformat/openwire/marshal/v4/ExceptionResponseMarshaller.h	
3408	
src/main/activemq/wireformat/openwire/marshal/v4/FlushCommandMarshaller.h	3412
src/main/activemq/wireformat/openwire/marshal/v4/IntegerResponseMarshaller.h	
3415	
src/main/activemq/wireformat/openwire/marshal/v4/JournalQueueAckMarshaller.h	
3418	
src/main/activemq/wireformat/openwire/marshal/v4/JournalTopicAckMarshaller.h	
3422	
src/main/activemq/wireformat/openwire/marshal/v4/JournalTraceMarshaller.h	3425
src/main/activemq/wireformat/openwire/marshal/v4/JournalTransactionMarshaller.h	
3428	
src/main/activemq/wireformat/openwire/marshal/v4/KeepAliveInfoMarshaller.h	3432
src/main/activemq/wireformat/openwire/marshal/v4/LastPartialCommandMarshaller.h	
3435	
src/main/activemq/wireformat/openwire/marshal/v4/LocalTransactionIdMarshaller.h	
3438	
src/main/activemq/wireformat/openwire/marshal/v4/MarshallerFactory.h	3441
src/main/activemq/wireformat/openwire/marshal/v4/MessageAckMarshaller.h	3444
src/main/activemq/wireformat/openwire/marshal/v4/MessageDispatchMarshaller.h	
3447	
src/main/activemq/wireformat/openwire/marshal/v4/MessageDispatchNotificationMarshaller.h	
3451	

src/main/activemq/wireformat/openwire/marshall/v4/MessageIdMarshaller.h . . .	3454
src/main/activemq/wireformat/openwire/marshall/v4/MessageMarshaller.h	3458
src/main/activemq/wireformat/openwire/marshall/v4/MessagePullMarshaller.h . .	3461
src/main/activemq/wireformat/openwire/marshall/v4/NetworkBridgeFilterMarshaller.h	3464
src/main/activemq/wireformat/openwire/marshall/v4/PartialCommandMarshaller.h	3468
src/main/activemq/wireformat/openwire/marshall/v4/ProducerAckMarshaller.h . .	3471
src/main/activemq/wireformat/openwire/marshall/v4/ProducerIdMarshaller.h . . .	3474
src/main/activemq/wireformat/openwire/marshall/v4/ProducerInfoMarshaller.h . .	3478
src/main/activemq/wireformat/openwire/marshall/v4/RemoveInfoMarshaller.h . .	3481
src/main/activemq/wireformat/openwire/marshall/v4/RemoveSubscriptionInfoMarshaller.h	3484
src/main/activemq/wireformat/openwire/marshall/v4/ReplayCommandMarshaller.h	3488
src/main/activemq/wireformat/openwire/marshall/v4/ResponseMarshaller.h	3491
src/main/activemq/wireformat/openwire/marshall/v4/SessionIdMarshaller.h	3494
src/main/activemq/wireformat/openwire/marshall/v4/SessionInfoMarshaller.h . . .	3498
src/main/activemq/wireformat/openwire/marshall/v4/ShutdownInfoMarshaller.h .	3501
src/main/activemq/wireformat/openwire/marshall/v4/SubscriptionInfoMarshaller.h	3504
src/main/activemq/wireformat/openwire/marshall/v4/TransactionIdMarshaller.h .	3508
src/main/activemq/wireformat/openwire/marshall/v4/TransactionInfoMarshaller.h	3511
src/main/activemq/wireformat/openwire/marshall/v4/WireFormatInfoMarshaller.h	3514
src/main/activemq/wireformat/openwire/marshall/v4/XATransactionIdMarshaller.h	3518
src/main/activemq/wireformat/openwire/marshall/v5/ActiveMQBlobMessageMarshaller.h	3316
src/main/activemq/wireformat/openwire/marshall/v5/ActiveMQBytesMessageMarshaller.h	3319
src/main/activemq/wireformat/openwire/marshall/v5/ActiveMQDestinationMarshaller.h	3322
src/main/activemq/wireformat/openwire/marshall/v5/ActiveMQMapMessageMarshaller.h	3326
src/main/activemq/wireformat/openwire/marshall/v5/ActiveMQMessageMarshaller.h	3329
src/main/activemq/wireformat/openwire/marshall/v5/ActiveMQObjectMessageMarshaller.h	3332
src/main/activemq/wireformat/openwire/marshall/v5/ActiveMQQueueMarshaller.h	3336
src/main/activemq/wireformat/openwire/marshall/v5/ActiveMQStreamMessageMarshaller.h	3339
src/main/activemq/wireformat/openwire/marshall/v5/ActiveMQTempDestinationMarshaller.h	3342
src/main/activemq/wireformat/openwire/marshall/v5/ActiveMQTempQueueMarshaller.h	3346
src/main/activemq/wireformat/openwire/marshall/v5/ActiveMQTempTopicMarshaller.h	3349
src/main/activemq/wireformat/openwire/marshall/v5/ActiveMQTextMessageMarshaller.h	3352
src/main/activemq/wireformat/openwire/marshall/v5/ActiveMQTopicMarshaller.h	3356
src/main/activemq/wireformat/openwire/marshall/v5/BaseCommandMarshaller.h	3359
src/main/activemq/wireformat/openwire/marshall/v5/BrokerIdMarshaller.h	3362
src/main/activemq/wireformat/openwire/marshall/v5/BrokerInfoMarshaller.h . . .	3366

src/main/activemq/wireformat/openwire/marshal/v5/	ConnectionControlMarshaller.h	
	3369	
src/main/activemq/wireformat/openwire/marshal/v5/	ConnectionErrorMarshaller.h	
	3372	
src/main/activemq/wireformat/openwire/marshal/v5/	ConnectionIdMarshaller.h	3376
src/main/activemq/wireformat/openwire/marshal/v5/	ConnectionInfoMarshaller.h	3379
src/main/activemq/wireformat/openwire/marshal/v5/	ConsumerControlMarshaller.h	
	3382	
src/main/activemq/wireformat/openwire/marshal/v5/	ConsumerIdMarshaller.h	3386
src/main/activemq/wireformat/openwire/marshal/v5/	ConsumerInfoMarshaller.h	3389
src/main/activemq/wireformat/openwire/marshal/v5/	ControlCommandMarshaller.h	
	3392	
src/main/activemq/wireformat/openwire/marshal/v5/	DataArrayResponseMarshaller.h	
	3396	
src/main/activemq/wireformat/openwire/marshal/v5/	DataResponseMarshaller.h	3399
src/main/activemq/wireformat/openwire/marshal/v5/	DestinationInfoMarshaller.h	3402
src/main/activemq/wireformat/openwire/marshal/v5/	DiscoveryEventMarshaller.h	3406
src/main/activemq/wireformat/openwire/marshal/v5/	ExceptionResponseMarshaller.h	
	3409	
src/main/activemq/wireformat/openwire/marshal/v5/	FlushCommandMarshaller.h	3412
src/main/activemq/wireformat/openwire/marshal/v5/	IntegerResponseMarshaller.h	
	3416	
src/main/activemq/wireformat/openwire/marshal/v5/	JournalQueueAckMarshaller.h	
	3419	
src/main/activemq/wireformat/openwire/marshal/v5/	JournalTopicAckMarshaller.h	
	3422	
src/main/activemq/wireformat/openwire/marshal/v5/	JournalTraceMarshaller.h	3426
src/main/activemq/wireformat/openwire/marshal/v5/	JournalTransactionMarshaller.h	
	3429	
src/main/activemq/wireformat/openwire/marshal/v5/	KeepAliveInfoMarshaller.h	3432
src/main/activemq/wireformat/openwire/marshal/v5/	LastPartialCommandMarshaller.h	
	3436	
src/main/activemq/wireformat/openwire/marshal/v5/	LocalTransactionIdMarshaller.h	
	3439	
src/main/activemq/wireformat/openwire/marshal/v5/	MarshallerFactory.h	3442
src/main/activemq/wireformat/openwire/marshal/v5/	MessageAckMarshaller.h	3445
src/main/activemq/wireformat/openwire/marshal/v5/	MessageDispatchMarshaller.h	
	3448	
src/main/activemq/wireformat/openwire/marshal/v5/	MessageDispatchNotificationMarshaller.h	
	3452	
src/main/activemq/wireformat/openwire/marshal/v5/	MessageIdMarshaller.h	3455
src/main/activemq/wireformat/openwire/marshal/v5/	MessageMarshaller.h	3458
src/main/activemq/wireformat/openwire/marshal/v5/	MessagePullMarshaller.h	3462
src/main/activemq/wireformat/openwire/marshal/v5/	NetworkBridgeFilterMarshaller.h	
	3465	
src/main/activemq/wireformat/openwire/marshal/v5/	PartialCommandMarshaller.h	
	3468	
src/main/activemq/wireformat/openwire/marshal/v5/	ProducerAckMarshaller.h	3472
src/main/activemq/wireformat/openwire/marshal/v5/	ProducerIdMarshaller.h	3475
src/main/activemq/wireformat/openwire/marshal/v5/	ProducerInfoMarshaller.h	3478
src/main/activemq/wireformat/openwire/marshal/v5/	RemoveInfoMarshaller.h	3482
src/main/activemq/wireformat/openwire/marshal/v5/	RemoveSubscriptionInfoMarshaller.h	
	3485	

src/main/activemq/wireformat/openwire/marshall/v5/ReplayCommandMarshaller.h	
3488	
src/main/activemq/wireformat/openwire/marshall/v5/ResponseMarshaller.h	3492
src/main/activemq/wireformat/openwire/marshall/v5/SessionIdMarshaller.h	3495
src/main/activemq/wireformat/openwire/marshall/v5/SessionInfoMarshaller.h . . .	3498
src/main/activemq/wireformat/openwire/marshall/v5/ShutdownInfoMarshaller.h .	3502
src/main/activemq/wireformat/openwire/marshall/v5/SubscriptionInfoMarshaller.h	
3505	
src/main/activemq/wireformat/openwire/marshall/v5/TransactionIdMarshaller.h .	3508
src/main/activemq/wireformat/openwire/marshall/v5/TransactionInfoMarshaller.h	3512
src/main/activemq/wireformat/openwire/marshall/v5/WireFormatInfoMarshaller.h	3515
src/main/activemq/wireformat/openwire/marshall/v5/XATransactionIdMarshaller.h	
3518	
src/main/activemq/wireformat/openwire/utils/BooleanStream.h	3521
src/main/activemq/wireformat/openwire/utils/HexTable.h	3522
src/main/activemq/wireformat/openwire/utils/MessagePropertyInterceptor.h . . .	3523
src/main/activemq/wireformat/openwire/utils/OpenwireStringSupport.h	3523
src/main/activemq/wireformat/stomp/StompCommandConstants.h	3524
src/main/activemq/wireformat/stomp/StompFrame.h	3524
src/main/activemq/wireformat/stomp/StompHelper.h	3525
src/main/activemq/wireformat/stomp/StompWireFormat.h	3525
src/main/activemq/wireformat/stomp/StompWireFormatFactory.h	3526
src/main/cms/BytesMessage.h	3529
src/main/cms/Closeable.h	3529
src/main/cms/CMSException.h	3530
src/main/cms/CMSProperties.h	3531
src/main/cms/CMSSecurityException.h	3531
src/main/cms/Config.h	3306
src/main/cms/Connection.h	3532
src/main/cms/ConnectionFactory.h	3532
src/main/cms/ConnectionMetaData.h	3533
src/main/cms/DeliveryMode.h	3533
src/main/cms/Destination.h	3533
src/main/cms/ExceptionListener.h	3534
src/main/cms/IllegalStateException.h	3534
src/main/cms/InvalidClientIdException.h	3535
src/main/cms/InvalidDestinationException.h	3536
src/main/cms/InvalidSelectorException.h	3536
src/main/cms/MapMessage.h	3536
src/main/cms/Message.h	3249
src/main/cms/MessageConsumer.h	3537
src/main/cms/MessageEOFException.h	3537
src/main/cms/MessageFormatException.h	3538
src/main/cms/MessageListener.h	3538
src/main/cms/MessageNotReadableException.h	3539
src/main/cms/MessageNotWritableException.h	3539
src/main/cms/MessageProducer.h	3540
src/main/cms/ObjectMessage.h	3540
src/main/cms/Queue.h	3541
src/main/cms/QueueBrowser.h	3542
src/main/cms/Session.h	3542
src/main/cms/Startable.h	3543
src/main/cms/Stopable.h	3543
src/main/cms/StreamMessage.h	3544

src/main/cms/ TemporaryQueue.h	3544
src/main/cms/ TemporaryTopic.h	3545
src/main/cms/ TextMessage.h	3545
src/main/cms/ Topic.h	3546
src/main/cms/ UnsupportedOperationException.h	3546
src/main/decaf/internal/ AprPool.h	3547
src/main/decaf/internal/ DecafRuntime.h	3547
src/main/decaf/internal/io/ StandardErrorOutputStream.h	3548
src/main/decaf/internal/io/ StandardInputStream.h	3548
src/main/decaf/internal/io/ StandardOutputStream.h	3549
src/main/decaf/internal/net/ URIEncoderDecoder.h	3549
src/main/decaf/internal/net/ URIHelper.h	3550
src/main/decaf/internal/net/ URIType.h	3550
src/main/decaf/internal/nio/ BufferFactory.h	3551
src/main/decaf/internal/nio/ ByteBuffer.h	3551
src/main/decaf/internal/nio/ ByteBufferPerspective.h	3552
src/main/decaf/internal/nio/ CharArrayBuffer.h	3552
src/main/decaf/internal/nio/ DoubleArrayBuffer.h	3553
src/main/decaf/internal/nio/ FloatArrayBuffer.h	3554
src/main/decaf/internal/nio/ IntArrayBuffer.h	3554
src/main/decaf/internal/nio/ LongArrayBuffer.h	3555
src/main/decaf/internal/nio/ ShortArrayBuffer.h	3555
src/main/decaf/internal/util/ ByteArrayAdapter.h	3556
src/main/decaf/internal/util/ HexStringParser.h	3561
src/main/decaf/internal/util/ TimerTaskHeap.h	3561
src/main/decaf/internal/util/concurrent/ ConditionImpl.h	3556
src/main/decaf/internal/util/concurrent/ MutexImpl.h	3557
src/main/decaf/internal/util/concurrent/ SynchronizableImpl.h	3557
src/main/decaf/internal/util/concurrent/ Transferer.h	3558
src/main/decaf/internal/util/concurrent/ TransferQueue.h	3558
src/main/decaf/internal/util/concurrent/ TransferStack.h	3559
src/main/decaf/internal/util/concurrent/unix/ ConditionHandle.h	3559
src/main/decaf/internal/util/concurrent/unix/ MutexHandle.h	3560
src/main/decaf/internal/util/concurrent/windows/ ConditionHandle.h	3560
src/main/decaf/internal/util/concurrent/windows/ MutexHandle.h	3561
src/main/decaf/io/ BlockingByteArrayInputStream.h	3562
src/main/decaf/io/ BufferedInputStream.h	3562
src/main/decaf/io/ BufferedOutputStream.h	3563
src/main/decaf/io/ ByteArrayInputStream.h	3563
src/main/decaf/io/ ByteArrayOutputStream.h	3564
src/main/decaf/io/ Closeable.h	3530
src/main/decaf/io/ DataInputStream.h	3564
src/main/decaf/io/ DataOutputStream.h	3565
src/main/decaf/io/ EOFException.h	3565
src/main/decaf/io/ FilterInputStream.h	3566
src/main/decaf/io/ FilterOutputStream.h	3566
src/main/decaf/io/ InputStream.h	3567
src/main/decaf/io/ InterruptedIOException.h	3567
src/main/decaf/io/ IOException.h	3568
src/main/decaf/io/ OutputStream.h	3568
src/main/decaf/io/ Reader.h	3569
src/main/decaf/io/ UnsupportedEncodingException.h	3569
src/main/decaf/io/ UTFDataFormatException.h	3569
src/main/decaf/io/ Writer.h	3570

src/main/decaf/lang/	Appendable.h	3570
src/main/decaf/lang/	Boolean.h	3571
src/main/decaf/lang/	Byte.h	3571
src/main/decaf/lang/	Character.h	3572
src/main/decaf/lang/	CharSequence.h	3572
src/main/decaf/lang/	Comparable.h	3572
src/main/decaf/lang/	Double.h	3573
src/main/decaf/lang/	Exception.h	3573
src/main/decaf/lang/	Float.h	3578
src/main/decaf/lang/	Integer.h	3579
src/main/decaf/lang/	Iterable.h	3579
src/main/decaf/lang/	Long.h	3579
src/main/decaf/lang/	Math.h	3580
src/main/decaf/lang/	Number.h	3580
src/main/decaf/lang/	Pointer.h	3581
src/main/decaf/lang/	Runnable.h	3582
src/main/decaf/lang/	Runtime.h	3582
src/main/decaf/lang/	Short.h	3583
src/main/decaf/lang/	System.h	3583
src/main/decaf/lang/	Thread.h	3583
src/main/decaf/lang/	ThreadGroup.h	3584
src/main/decaf/lang/	Throwable.h	3584
src/main/decaf/lang/exceptions/	ClassCastException.h	3574
src/main/decaf/lang/exceptions/	ExceptionDefines.h	3274
src/main/decaf/lang/exceptions/	IllegalArgumentException.h	3574
src/main/decaf/lang/exceptions/	IllegalMonitorStateException.h	3575
src/main/decaf/lang/exceptions/	IllegalStateException.h	3535
src/main/decaf/lang/exceptions/	IllegalThreadStateException.h	3575
src/main/decaf/lang/exceptions/	IndexOutOfBoundsException.h	3575
src/main/decaf/lang/exceptions/	InterruptedException.h	3576
src/main/decaf/lang/exceptions/	InvalidStateException.h	3576
src/main/decaf/lang/exceptions/	NoSuchElementException.h	3577
src/main/decaf/lang/exceptions/	NullPointerException.h	3577
src/main/decaf/lang/exceptions/	NumberFormatException.h	3577
src/main/decaf/lang/exceptions/	RuntimeException.h	3578
src/main/decaf/lang/exceptions/	UnsupportedOperationException.h	3546
src/main/decaf/net/	BindException.h	3585
src/main/decaf/net/	BufferedSocket.h	3585
src/main/decaf/net/	ConnectException.h	3586
src/main/decaf/net/	HttpRetryException.h	3586
src/main/decaf/net/	MalformedURLException.h	3587
src/main/decaf/net/	NoRouteToHostException.h	3587
src/main/decaf/net/	PortUnreachableException.h	3587
src/main/decaf/net/	ProtocolException.h	3588
src/main/decaf/net/	ServerSocket.h	3588
src/main/decaf/net/	Socket.h	3589
src/main/decaf/net/	SocketError.h	3589
src/main/decaf/net/	SocketException.h	3590
src/main/decaf/net/	SocketFactory.h	3590
src/main/decaf/net/	SocketInputStream.h	3591
src/main/decaf/net/	SocketOutputStream.h	3591
src/main/decaf/net/	SocketTimeoutException.h	3592
src/main/decaf/net/	TcpSocket.h	3592
src/main/decaf/net/	UnknownHostException.h	3593

src/main/decaf/net/UnknownServiceException.h	3593
src/main/decaf/net/URI.h	3593
src/main/decaf/net/URISyntaxException.h	3594
src/main/decaf/net/URL.h	3594
src/main/decaf/net/URLDecoder.h	3595
src/main/decaf/net/URLEncoder.h	3595
src/main/decaf/nio/Buffer.h	3596
src/main/decaf/nio/BufferOverflowException.h	3596
src/main/decaf/nio/BufferUnderflowException.h	3596
src/main/decaf/nio/ByteBuffer.h	3597
src/main/decaf/nio/CharBuffer.h	3597
src/main/decaf/nio/DoubleBuffer.h	3598
src/main/decaf/nio/FloatBuffer.h	3599
src/main/decaf/nio/IntBuffer.h	3599
src/main/decaf/nio/InvalidMarkException.h	3600
src/main/decaf/nio/LongBuffer.h	3600
src/main/decaf/nio/ReadOnlyBufferException.h	3601
src/main/decaf/nio/ShortBuffer.h	3601
src/main/decaf/security/GeneralSecurityException.h	3605
src/main/decaf/security/InvalidKeyException.h	3606
src/main/decaf/security/Key.h	3606
src/main/decaf/security/KeyException.h	3607
src/main/decaf/security/NoSuchAlgorithmException.h	3607
src/main/decaf/security/NoSuchProviderException.h	3607
src/main/decaf/security/Principal.h	3608
src/main/decaf/security/PublicKey.h	3608
src/main/decaf/security/SignatureException.h	3609
src/main/decaf/security/auth/x500/X500Principal.h	3602
src/main/decaf/security/cert/Certificate.h	3602
src/main/decaf/security/cert/CertificateEncodingException.h	3603
src/main/decaf/security/cert/CertificateException.h	3603
src/main/decaf/security/cert/CertificateExpiredException.h	3604
src/main/decaf/security/cert/CertificateNotYetValidException.h	3604
src/main/decaf/security/cert/CertificateParsingException.h	3604
src/main/decaf/security/cert/X509Certificate.h	3605
src/main/decaf/security_provider/SecurityProvider.h	3609
src/main/decaf/security_provider/SecurityProviderMap.h	3609
src/main/decaf/security_provider/SecurityProviderRegistrar.h	3610
src/main/decaf/security_provider/unix/openssl/OpenSSLX500Principal.h	3610
src/main/decaf/security_provider/unix/openssl/OpenSSLX509Certificate.h	3611
src/main/decaf/util/AbstractCollection.h	3611
src/main/decaf/util/AbstractList.h	3612
src/main/decaf/util/AbstractMap.h	3612
src/main/decaf/util/AbstractQueue.h	3613
src/main/decaf/util/AbstractSequentialList.h	3614
src/main/decaf/util/AbstractSet.h	3614
src/main/decaf/util/Collection.h	3615
src/main/decaf/util/Comparator.h	3615
src/main/decaf/util/Config.h	3306
src/main/decaf/util/Date.h	3634
src/main/decaf/util/Iterator.h	3635
src/main/decaf/util/List.h	3635
src/main/decaf/util/ListIterator.h	3636
src/main/decaf/util/Map.h	3644

src/main/decaf/util/	PriorityQueue.h	3645
src/main/decaf/util/	Properties.h	3645
src/main/decaf/util/	Queue.h	3541
src/main/decaf/util/	Random.h	3646
src/main/decaf/util/	Set.h	3647
src/main/decaf/util/	StlList.h	3647
src/main/decaf/util/	StlMap.h	3648
src/main/decaf/util/	StlQueue.h	3648
src/main/decaf/util/	StlSet.h	3649
src/main/decaf/util/	StringTokenizer.h	3650
src/main/decaf/util/	Timer.h	3650
src/main/decaf/util/	TimerTask.h	3651
src/main/decaf/util/	UUID.h	3651
src/main/decaf/util/comparators/	Less.h	3616
src/main/decaf/util/concurrent/	BlockingQueue.h	3618
src/main/decaf/util/concurrent/	BrokenBarrierException.h	3618
src/main/decaf/util/concurrent/	Callable.h	3619
src/main/decaf/util/concurrent/	CancellationException.h	3619
src/main/decaf/util/concurrent/	Concurrent.h	3619
src/main/decaf/util/concurrent/	ConcurrentMap.h	3620
src/main/decaf/util/concurrent/	ConcurrentStlMap.h	3621
src/main/decaf/util/concurrent/	CountDownLatch.h	3622
src/main/decaf/util/concurrent/	Delayed.h	3622
src/main/decaf/util/concurrent/	ExecutionException.h	3623
src/main/decaf/util/concurrent/	Executor.h	3623
src/main/decaf/util/concurrent/	ExecutorService.h	3623
src/main/decaf/util/concurrent/	Future.h	3624
src/main/decaf/util/concurrent/	Lock.h	3624
src/main/decaf/util/concurrent/	Mutex.h	3628
src/main/decaf/util/concurrent/	PooledThread.h	3628
src/main/decaf/util/concurrent/	PooledThreadListener.h	3629
src/main/decaf/util/concurrent/	RejectedExecutionException.h	3629
src/main/decaf/util/concurrent/	RejectedExecutionHandler.h	3629
src/main/decaf/util/concurrent/	Semaphore.h	3630
src/main/decaf/util/concurrent/	Synchronizable.h	3630
src/main/decaf/util/concurrent/	SynchronousQueue.h	3631
src/main/decaf/util/concurrent/	TaskListener.h	3632
src/main/decaf/util/concurrent/	ThreadFactory.h	3632
src/main/decaf/util/concurrent/	ThreadPool.h	3632
src/main/decaf/util/concurrent/	TimeoutException.h	3633
src/main/decaf/util/concurrent/	TimeUnit.h	3634
src/main/decaf/util/concurrent/atomic/	AtomicBoolean.h	3616
src/main/decaf/util/concurrent/atomic/	AtomicInteger.h	3617
src/main/decaf/util/concurrent/atomic/	AtomicReference.h	3617
src/main/decaf/util/concurrent/locks/	Condition.h	3625
src/main/decaf/util/concurrent/locks/	Lock.h	3625
src/main/decaf/util/concurrent/locks/	LockSupport.h	3626
src/main/decaf/util/concurrent/locks/	ReadWriteLock.h	3627
src/main/decaf/util/concurrent/locks/	ReentrantLock.h	3627
src/main/decaf/util/logging/	ConsoleHandler.h	3636
src/main/decaf/util/logging/	Filter.h	3636
src/main/decaf/util/logging/	Formatter.h	3637
src/main/decaf/util/logging/	Handler.h	3637
src/main/decaf/util/logging/	Logger.h	3638

src/main/decaf/util/logging/ LoggerCommon.h	3638
src/main/decaf/util/logging/ LoggerDefines.h	3639
src/main/decaf/util/logging/ LoggerHierarchy.h	3640
src/main/decaf/util/logging/ LogManager.h	3640
src/main/decaf/util/logging/ LogRecord.h	3641
src/main/decaf/util/logging/ LogWriter.h	3641
src/main/decaf/util/logging/ MarkBlockLogger.h	3642
src/main/decaf/util/logging/ PropertiesChangeListener.h	3642
src/main/decaf/util/logging/ SimpleFormatter.h	3643
src/main/decaf/util/logging/ SimpleLogger.h	3643
src/main/decaf/util/logging/ StreamHandler.h	3644

Chapter 5

Namespace Documentation

5.1 activemq Namespace Reference

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

Namespaces

- namespace **cmsutil**
- namespace **commands**
- namespace **core**
- namespace **exceptions**
- namespace **io**
- namespace **library**
- namespace **state**
- namespace **threads**
- namespace **transport**
- namespace **util**
- namespace **wireformat**

5.1.1 Detailed Description

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements. See the NOTICE file distributed with this work for additional information regarding copyright ownership. The ASF licenses this file to You under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

5.2 activemq::cmsutil Namespace Reference

Data Structures

- class **CachedConsumer**

A cached message consumer contained within a pooled session.

- class **CachedProducer**

A cached message producer contained within a pooled session.

- class **CmsAccessor**

*Base class for **activemq.cmsutil.CmsTemplate** (p. 969) and other CMS-accessing gateway helpers, defining common properties such as the CMS **cms.ConnectionFactory** (p. 1103) to operate on.*

- class **CmsDestinationAccessor**

*Extends the **CmsAccessor** (p. 954) to add support for resolving destination names.*

- class **CmsTemplate**

***CmsTemplate** (p. 969) simplifies performing synchronous CMS operations.*

- class **DestinationResolver**

*Resolves a CMS destination name to a **Destination**.*

- class **DynamicDestinationResolver**

*Resolves a CMS destination name to a **Destination**.*

- class **MessageCreator**

*Creates the user-defined message to be sent by the **CmsTemplate** (p. 969).*

- class **PooledSession**

A pooled session object that wraps around a delegate session.

- class **ProducerCallback**

Callback for sending a message to a CMS destination.

- class **ResourceLifecycleManager**

Manages the lifecycle of a set of CMS resources.

- class **SessionCallback**

Callback for executing any number of operations on a provided CMS Session.

- class **SessionPool**

A pool of CMS sessions from the same connection and with the same acknowledge mode.

5.3 activemq::commands Namespace Reference

Data Structures

- class **ActiveMQBlobMessage**
- class **ActiveMQBytesMessage**
- class **ActiveMQDestination**
- class **ActiveMQMapMessage**
- class **ActiveMQMessage**
- class **ActiveMQMessageTemplate**
- class **ActiveMQObjectMessage**
- class **ActiveMQQueue**
- class **ActiveMQStreamMessage**
- class **ActiveMQTempDestination**
- class **ActiveMQTempQueue**
- class **ActiveMQTempTopic**
- class **ActiveMQTextMessage**
- class **ActiveMQTopic**
- class **BaseCommand**
- class **BaseDataStructure**
- class **BooleanExpression**
- class **BrokerError**

This class represents an Exception sent from the Broker.

- class **BrokerId**
- class **BrokerInfo**
- class **Command**
- class **ConnectionControl**
- class **ConnectionError**
- class **ConnectionId**
- class **ConnectionInfo**
- class **ConsumerControl**
- class **ConsumerId**
- class **ConsumerInfo**
- class **ControlCommand**
- class **DataArrayResponse**
- class **DataResponse**
- class **DataStructure**
- class **DestinationInfo**
- class **DiscoveryEvent**
- class **ExceptionResponse**
- class **FlushCommand**
- class **IntegerResponse**
- class **JournalQueueAck**
- class **JournalTopicAck**
- class **JournalTrace**
- class **JournalTransaction**
- class **KeepAliveInfo**
- class **LastPartialCommand**
- class **LocalTransactionId**

- class **Message**
- class **MessageAck**
- class **MessageDispatch**
- class **MessageDispatchNotification**
- class **MessageId**
- class **MessagePull**
- class **NetworkBridgeFilter**
- class **PartialCommand**
- class **ProducerAck**
- class **ProducerId**
- class **ProducerInfo**
- class **RemoveInfo**
- class **RemoveSubscriptionInfo**
- class **ReplayCommand**
- class **Response**
- class **SessionId**
- class **SessionInfo**
- class **ShutdownInfo**
- class **SubscriptionInfo**
- class **TransactionId**
- class **TransactionInfo**
- class **WireFormatInfo**
- class **XATransactionId**

5.4 activemq::core Namespace Reference

Data Structures

- class **ActiveMQAckHandler**
Interface class that is used to give CMS Messages an interface to Ack themselves with.
- class **ActiveMQConnection**
Concrete connection used for all connectors to the ActiveMQ broker.
- class **ActiveMQConnectionFactory**
- class **ActiveMQConnectionMetaData**
*This class houses all the various settings and information that is used by an instance of an **ActiveMQConnection** (p. 202) class.*
- class **ActiveMQConnectionSupport**
- class **ActiveMQConstants**
Class holding constant values for various ActiveMQ specific things Each constant is defined as an enumeration and has functions that convert back an forth between string and enum values.
- class **ActiveMQConsumer**
- class **ActiveMQProducer**
- class **ActiveMQSession**
- class **ActiveMQSessionExecutor**
Delegate dispatcher for a single session.

- class **ActiveMQTransactionContext**
Transaction Management class, hold messages that are to be redelivered upon a request to roll-back.
- class **DispatchData**
Simple POCO that contains the information necessary to route a message to a specified consumer.
- class **Dispatcher**
Interface for an object responsible for dispatching messages to consumers.
- class **MessageDispatchChannel**
- class **Synchronization**
*Transacted Object **Synchronization** (p. 2945), used to sync the events of a Transaction with the items in the Transaction.*

5.5 activemq::exceptions Namespace Reference

Data Structures

- class **ActiveMQException**
- class **BrokerException**

5.6 activemq::io Namespace Reference

Data Structures

- class **LoggingInputStream**
- class **LoggingOutputStream**
OutputStream filter that just logs the data being written.

5.7 activemq::library Namespace Reference

Data Structures

- class **ActiveMQCPP**

5.8 activemq::state Namespace Reference

Data Structures

- class **CommandVisitor**
Interface for an Object that can visit the various Command Objects that are sent from and to this client.

- class **CommandVisitorAdapter**

*Default Implementation of a **CommandVisitor** (p. 996) that returns NULL for all calls.*

- class **ConnectionState**
- class **ConnectionStateTracker**
- class **ConsumerState**
- class **ProducerState**
- class **SessionState**
- class **Tracked**
- class **TransactionState**

5.9 activemq::threads Namespace Reference

Data Structures

- class **CompositeTask**

*Represents a single task that can be part of a set of Tasks that are contained in a **CompositeTaskRunner** (p. 1017).*

- class **CompositeTaskRunner**

*A **Task** (p. 2956) Runner that can contain one or more CompositeTasks that are each checked for pending work and run if any is present in the order that the tasks were added.*

- class **DedicatedTaskRunner**
- class **Task**

Represents a unit of work that requires one or more iterations to complete.

- class **TaskRunner**

5.10 activemq::transport Namespace Reference

Namespaces

- namespace **correlator**
- namespace **failover**
- namespace **inactivity**
- namespace **logging**
- namespace **mock**
- namespace **tcp**

Data Structures

- class **AbstractTransportFactory**

*Abstract implementation of the **TransportFactory** (p. 3071) interface, providing the base functionality that's common to most of the **TransportFactory** (p. 3071) instances.*

- class **CompositeTransport**

*A Composite **Transport** (p. 3066) is a **Transport** (p. 3066) implementation that is composed of several Transports.*

- class **DefaultTransportListener**
- class **IOTransport**

*Implementation of the **Transport** (p. 3066) interface that performs marshaling of commands to IO streams.*

- class **Transport**

Interface for a transport layer for command objects.

- class **TransportFactory**

Defines the interface for Factories that create Transports or TransportFilters.

- class **TransportFilter**

A filter on the transport layer.

- class **TransportListener**

A listener of asynchronous exceptions from a command transport object.

- class **TransportRegistry**

*Registry of all **Transport** (p. 3066) Factories that are available to the client at runtime.*

5.11 activemq::transport::correlator Namespace Reference

Data Structures

- class **FutureResponse**

A container that holds a response object.

- class **ResponseCorrelator**

This type of transport filter is responsible for correlating asynchronous responses with requests.

5.12 activemq::transport::failover Namespace Reference

Data Structures

- class **BackupTransport**
- class **BackupTransportPool**
- class **CloseTransportsTask**
- class **FailoverTransport**
- class **FailoverTransportFactory**

*Creates an instance of a **FailoverTransport** (p. 1512).*

- class **FailoverTransportListener**

*Utility class used by the **Transport** (p. 3066) to perform the work of responding to events from the active **Transport** (p. 3066).*

- class **URIPool**

5.13 activemq::transport::inactivity Namespace Reference

Data Structures

- class **InactivityMonitor**
- class **ReadChecker**
Runnable class that is used by the {}.
- class **WriteChecker**
Runnable class used by the {}.

5.14 activemq::transport::logging Namespace Reference

Data Structures

- class **LoggingTransport**
A transport filter that logs commands as they are sent/received.

5.15 activemq::transport::mock Namespace Reference

Data Structures

- class **InternalCommandListener**
*Listens for Commands sent from the **MockTransport** (p. 2227).*
- class **MockTransport**
*The **MockTransport** (p. 2227) defines a base level **Transport** (p. 3066) class that is intended to be used in place of an a regular protocol **Transport** (p. 3066) such as TCP.*
- class **MockTransportFactory**
Manufactures MockTransports, which are objects that read from input streams and write to output streams.
- class **ResponseBuilder**
Interface for all Protocols to implement that defines the behavior of the Broker in response to messages of that protocol.

5.16 activemq::transport::tcp Namespace Reference

Data Structures

- class **TcpTransport**

*Implements a TCP/IP based transport filter, this transport is meant to wrap an instance of an **IOTransport** (p. 1709).*

- class **TcpTransportFactory**

*Factory Responsible for creating the **TcpTransport** (p. 2967).*

5.17 activemq::util Namespace Reference

Data Structures

- class **ActiveMQProperties**

*Implementation of the **CMSProperties** interface that delegates to a **decaf::util::Properties** (p. 2494) object.*

- class **CMSExceptionSupport**

- class **CompositeData**

Represents a Composite URI.

- class **LongSequenceGenerator**

This class is used to generate a sequence of long long values that are incremented each time a new value is requested.

- class **MemoryUsage**

- class **PrimitiveList**

List of primitives.

- class **PrimitiveMap**

Map of named primitives.

- class **PrimitiveValueConverter**

*Class controls the conversion of data contained in a **PrimitiveValueNode** (p. 2398) from one type to another.*

- class **PrimitiveValueNode**

Class that wraps around a single value of one of the many types.

- class **URISupport**

- class **Usage**

5.18 activemq::wireformat Namespace Reference

Namespaces

- namespace **openwire**
- namespace **stomp**

Data Structures

- class **MarshalAware**
- class **WireFormat**
Provides a mechanism to marshal commands into and out of packets or into and out of streams, Channels and Datagrams.
- class **WireFormatFactory**
*The **WireFormatFactory** (p. 3151) is the interface that all **WireFormatFactory** (p. 3151) classes must extend.*
- class **WireFormatNegotiator**
*Defines a **WireFormatNegotiator** (p. 3182) which allows a **WireFormat** (p. 3148) to.*
- class **WireFormatRegistry**
*Registry of all **WireFormat** (p. 3148) Factories that are available to the client at runtime.*

5.19 activemq::wireformat::openwire Namespace Reference

Namespaces

- namespace **marshal**
- namespace **utils**

Data Structures

- class **OpenWireFormat**
- class **OpenWireFormatFactory**
- class **OpenWireFormatNegotiator**
- class **OpenWireResponseBuilder**

5.20 activemq::wireformat::openwire::marshal Namespace Reference

Namespaces

- namespace **v1**
- namespace **v2**
- namespace **v3**
- namespace **v4**
- namespace **v5**

Data Structures

- class **BaseDataStreamMarshaller**
Base class for all Marshallers that marshal DataStructures to and from the wire using the Open-Wire protocol.
- class **DataStreamMarshaller**
Base class for all classes that marshal commands for Openwire.
- class **PrimitiveTypesMarshaller**
This class wraps the functionality needed to marshal a primitive map to the Openwire Format's expectation of what the map looks like on the wire.

5.21 activemq::wireformat::openwire::marshal::v1 Namespace Reference

Data Structures

- class **ActiveMQBlobMessageMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 150).*
- class **ActiveMQBytesMessageMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 186).*
- class **ActiveMQDestinationMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 254).*
- class **ActiveMQMapMessageMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 288).*
- class **ActiveMQMessageMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 311).*
- class **ActiveMQObjectMessageMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 351).*
- class **ActiveMQQueueMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 386).*
- class **ActiveMQStreamMessageMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 439).*
- class **ActiveMQTempDestinationMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 462).*
- class **ActiveMQTempQueueMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 485).*

- class **ActiveMQTempTopicMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 509).*
- class **ActiveMQTextMessageMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 533).*
- class **ActiveMQTopicMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 557).*
- class **BaseCommandMarshaller**
*Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 610).*
- class **BrokerIdMarshaller**
*Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 698).*
- class **BrokerInfoMarshaller**
*Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 725).*
- class **ConnectionControlMarshaller**
*Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1063).*
- class **ConnectionErrorMarshaller**
*Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1087).*
- class **ConnectionIdMarshaller**
*Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1113).*
- class **ConnectionInfoMarshaller**
*Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1142).*
- class **ConsumerControlMarshaller**
*Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1178).*
- class **ConsumerIdMarshaller**
*Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1202).*
- class **ConsumerInfoMarshaller**
*Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1226).*
- class **ControlCommandMarshaller**
*Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1254).*
- class **DataArrayResponseMarshaller**
*Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1279).*
- class **DataResponseMarshaller**
*Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1314).*
- class **DestinationInfoMarshaller**

*Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1407).*

- class **DiscoveryEventMarshaller**

*Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1435).*

- class **ExceptionResponseMarshaller**

*Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1491).*

- class **FlushCommandMarshaller**

*Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 1579).*

- class **IntegerResponseMarshaller**

*Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 1673).*

- class **JournalQueueAckMarshaller**

*Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 1733).*

- class **JournalTopicAckMarshaller**

*Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 1750).*

- class **JournalTraceMarshaller**

*Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 1781).*

- class **JournalTransactionMarshaller**

*Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 1804).*

- class **KeepAliveInfoMarshaller**

*Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 1831).*

- class **LastPartialCommandMarshaller**

*Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 1846).*

- class **LocalTransactionIdMarshaller**

*Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 1890).*

- class **MarshallerFactory**

Used to create marshallers for a specific version of the wire protocol.

- class **MessageAckMarshaller**

*Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2076).*

- class **MessageDispatchMarshaller**

*Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2107).*

- class **MessageDispatchNotificationMarshaller**

*Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 2132).*

- class **MessageIdMarshaller**

*Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2161).*

- class **MessageMarshaller**
*Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2183).*
- class **MessagePullMarshaller**
*Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2215).*
- class **NetworkBridgeFilterMarshaller**
*Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2261).*
- class **PartialCommandMarshaller**
*Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2335).*
- class **ProducerAckMarshaller**
*Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 2439).*
- class **ProducerIdMarshaller**
*Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 2461).*
- class **ProducerInfoMarshaller**
*Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 2490).*
- class **RemoveInfoMarshaller**
*Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 2540).*
- class **RemoveSubscriptionInfoMarshaller**
*Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 2572).*
- class **ReplayCommandMarshaller**
*Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 2603).*
- class **ResponseMarshaller**
*Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 2633).*
- class **SessionIdMarshaller**
*Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 2685).*
- class **SessionInfoMarshaller**
*Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 2709).*
- class **ShutdownInfoMarshaller**
*Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 2760).*
- class **SubscriptionInfoMarshaller**
*Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 2911).*
- class **TransactionIdMarshaller**
*Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3022).*
- class **TransactionInfoMarshaller**

*Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3048).*

- class **WireFormatInfoMarshaller**

*Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3170).*

- class **XATransactionIdMarshaller**

*Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3209).*

5.22 activemq::wireformat::openwire::marshal::v2 Namespace Reference

Data Structures

- class **ActiveMQBlobMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 162).*

- class **ActiveMQBytesMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 198).*

- class **ActiveMQDestinationMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 265).*

- class **ActiveMQMapMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 300).*

- class **ActiveMQMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 323).*

- class **ActiveMQObjectMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 363).*

- class **ActiveMQQueueMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 398).*

- class **ActiveMQStreamMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 451).*

- class **ActiveMQTempDestinationMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 473).*

- class **ActiveMQTempQueueMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 497).*

- class **ActiveMQTempTopicMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 521).*

- class **ActiveMQTextMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 545).*

- class **ActiveMQTopicMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 569).*
- class **BaseCommandMarshaller**
*Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 630).*
- class **BrokerIdMarshaller**
*Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 709).*
- class **BrokerInfoMarshaller**
*Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 737).*
- class **ConnectionControlMarshaller**
*Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1075).*
- class **ConnectionErrorMarshaller**
*Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1099).*
- class **ConnectionIdMarshaller**
*Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1124).*
- class **ConnectionInfoMarshaller**
*Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1150).*
- class **ConsumerControlMarshaller**
*Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1186).*
- class **ConsumerIdMarshaller**
*Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1210).*
- class **ConsumerInfoMarshaller**
*Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1238).*
- class **ControlCommandMarshaller**
*Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1262).*
- class **DataArrayResponseMarshaller**
*Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1287).*
- class **DataResponseMarshaller**
*Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1326).*
- class **DestinationInfoMarshaller**
*Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1395).*
- class **DiscoveryEventMarshaller**
*Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1420).*

- class **ExceptionResponseMarshaller**
*Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1487).*
- class **FlushCommandMarshaller**
*Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 1576).*
- class **IntegerResponseMarshaller**
*Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 1669).*
- class **JournalQueueAckMarshaller**
*Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 1722).*
- class **JournalTopicAckMarshaller**
*Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 1746).*
- class **JournalTraceMarshaller**
*Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 1769).*
- class **JournalTransactionMarshaller**
*Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 1793).*
- class **KeepAliveInfoMarshaller**
*Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 1815).*
- class **LastPartialCommandMarshaller**
*Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 1842).*
- class **LocalTransactionIdMarshaller**
*Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 1878).*
- class **MarshallerFactory**
Used to create marshallers for a specific version of the wire protocol.
- class **MessageAckMarshaller**
*Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2060).*
- class **MessageDispatchMarshaller**
*Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2095).*
- class **MessageDispatchNotificationMarshaller**
*Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 2120).*
- class **MessageIdMarshaller**
*Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2146).*
- class **MessageMarshaller**
*Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2166).*
- class **MessagePullMarshaller**

*Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2207).*

- class **NetworkBridgeFilterMarshaller**
*Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2250).*
- class **PartialCommandMarshaller**
*Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2322).*
- class **ProducerAckMarshaller**
*Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 2424).*
- class **ProducerIdMarshaller**
*Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 2450).*
- class **ProducerInfoMarshaller**
*Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 2474).*
- class **RemoveInfoMarshaller**
*Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 2548).*
- class **RemoveSubscriptionInfoMarshaller**
*Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 2568).*
- class **ReplayCommandMarshaller**
*Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 2587).*
- class **ResponseMarshaller**
*Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 2620).*
- class **SessionIdMarshaller**
*Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 2689).*
- class **SessionInfoMarshaller**
*Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 2705).*
- class **ShutdownInfoMarshaller**
*Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 2771).*
- class **SubscriptionInfoMarshaller**
*Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 2926).*
- class **TransactionIdMarshaller**
*Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3018).*
- class **TransactionInfoMarshaller**
*Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3056).*
- class **WireFormatInfoMarshaller**
*Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3163).*

- class **XATransactionIdMarshaller**

*Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3197).*

5.23 activemq::wireformat::openwire::marshal::v3 Namespace Reference

Data Structures

- class **ActiveMQBlobMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 146).*

- class **ActiveMQBytesMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 182).*

- class **ActiveMQDestinationMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 250).*

- class **ActiveMQMapMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 284).*

- class **ActiveMQMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 307).*

- class **ActiveMQObjectMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 347).*

- class **ActiveMQQueueMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 382).*

- class **ActiveMQStreamMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 436).*

- class **ActiveMQTempDestinationMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 459).*

- class **ActiveMQTempQueueMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 481).*

- class **ActiveMQTempTopicMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 505).*

- class **ActiveMQTextMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 529).*

- class **ActiveMQTopicMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 553).*

- class **BaseCommandMarshaller**
*Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 603).*
- class **BrokerIdMarshaller**
*Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 694).*
- class **BrokerInfoMarshaller**
*Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 721).*
- class **ConnectionControlMarshaller**
*Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1059).*
- class **ConnectionErrorMarshaller**
*Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1083).*
- class **ConnectionIdMarshaller**
*Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1109).*
- class **ConnectionInfoMarshaller**
*Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1134).*
- class **ConsumerControlMarshaller**
*Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1170).*
- class **ConsumerIdMarshaller**
*Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1195).*
- class **ConsumerInfoMarshaller**
*Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1223).*
- class **ControlCommandMarshaller**
*Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1246).*
- class **DataArrayResponseMarshaller**
*Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1271).*
- class **DataResponseMarshaller**
*Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1310).*
- class **DestinationInfoMarshaller**
*Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1399).*
- class **DiscoveryEventMarshaller**
*Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1424).*
- class **ExceptionResponseMarshaller**
*Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1495).*
- class **FlushCommandMarshaller**
*Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 1583).*

- class **IntegerResponseMarshaller**
*Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 1677).*
- class **JournalQueueAckMarshaller**
*Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 1729).*
- class **JournalTopicAckMarshaller**
*Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 1754).*
- class **JournalTraceMarshaller**
*Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 1777).*
- class **JournalTransactionMarshaller**
*Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 1797).*
- class **KeepAliveInfoMarshaller**
*Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 1823).*
- class **LastPartialCommandMarshaller**
*Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 1850).*
- class **LocalTransactionIdMarshaller**
*Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 1882).*
- class **MarshallerFactory**
Used to create marshallers for a specific version of the wire protocol.
- class **MessageAckMarshaller**
*Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2068).*
- class **MessageDispatchMarshaller**
*Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2103).*
- class **MessageDispatchNotificationMarshaller**
*Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 2128).*
- class **MessageIdMarshaller**
*Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2154).*
- class **MessageMarshaller**
*Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2175).*
- class **MessagePullMarshaller**
*Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2211).*
- class **NetworkBridgeFilterMarshaller**
*Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2253).*

- class **PartialCommandMarshaller**
*Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2326).*
- class **ProducerAckMarshaller**
*Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 2427).*
- class **ProducerIdMarshaller**
*Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 2454).*
- class **ProducerInfoMarshaller**
*Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 2478).*
- class **RemoveInfoMarshaller**
*Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 2552).*
- class **RemoveSubscriptionInfoMarshaller**
*Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 2576).*
- class **ReplayCommandMarshaller**
*Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 2591).*
- class **ResponseMarshaller**
*Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 2624).*
- class **SessionIdMarshaller**
*Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 2697).*
- class **SessionInfoMarshaller**
*Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 2721).*
- class **ShutdownInfoMarshaller**
*Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 2767).*
- class **SubscriptionInfoMarshaller**
*Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 2915).*
- class **TransactionIdMarshaller**
*Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3033).*
- class **TransactionInfoMarshaller**
*Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3044).*
- class **WireFormatInfoMarshaller**
*Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3178).*
- class **XATransactionIdMarshaller**
*Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3201).*

5.24 activemq::wireformat::openwire::marshal::v4 Namespace Reference

Data Structures

- class **ActiveMQBlobMessageMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 154).*
- class **ActiveMQBytesMessageMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 190).*
- class **ActiveMQDestinationMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 258).*
- class **ActiveMQMapMessageMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 292).*
- class **ActiveMQMessageMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 315).*
- class **ActiveMQObjectMessageMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 355).*
- class **ActiveMQQueueMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 390).*
- class **ActiveMQStreamMessageMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 443).*
- class **ActiveMQTempDestinationMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 466).*
- class **ActiveMQTempQueueMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 489).*
- class **ActiveMQTempTopicMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 513).*
- class **ActiveMQTextMessageMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 537).*
- class **ActiveMQTopicMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 561).*
- class **BaseCommandMarshaller**
*Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 616).*
- class **BrokerIdMarshaller**

*Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 701).*

- class **BrokerInfoMarshaller**

*Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 729).*

- class **ConnectionControlMarshaller**

*Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1067).*

- class **ConnectionErrorMarshaller**

*Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1091).*

- class **ConnectionIdMarshaller**

*Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1116).*

- class **ConnectionInfoMarshaller**

*Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1138).*

- class **ConsumerControlMarshaller**

*Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1174).*

- class **ConsumerIdMarshaller**

*Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1198).*

- class **ConsumerInfoMarshaller**

*Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1230).*

- class **ControlCommandMarshaller**

*Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1250).*

- class **DataArrayResponseMarshaller**

*Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1275).*

- class **DataResponseMarshaller**

*Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1322).*

- class **DestinationInfoMarshaller**

*Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1403).*

- class **DiscoveryEventMarshaller**

*Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1427).*

- class **ExceptionResponseMarshaller**

*Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1499).*

- class **FlushCommandMarshaller**

*Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 1587).*

- class **IntegerResponseMarshaller**

*Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 1681).*

- class **JournalQueueAckMarshaller**
*Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 1725).*
- class **JournalTopicAckMarshaller**
*Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 1758).*
- class **JournalTraceMarshaller**
*Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 1773).*
- class **JournalTransactionMarshaller**
*Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 1801).*
- class **KeepAliveInfoMarshaller**
*Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 1827).*
- class **LastPartialCommandMarshaller**
*Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 1854).*
- class **LocalTransactionIdMarshaller**
*Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 1886).*
- class **MarshallerFactory**
Used to create marshallers for a specific version of the wire protocol.
- class **MessageAckMarshaller**
*Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2072).*
- class **MessageDispatchMarshaller**
*Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2099).*
- class **MessageDispatchNotificationMarshaller**
*Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 2124).*
- class **MessageIdMarshaller**
*Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2150).*
- class **MessageMarshaller**
*Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2170).*
- class **MessagePullMarshaller**
*Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2223).*
- class **NetworkBridgeFilterMarshaller**
*Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2265).*
- class **PartialCommandMarshaller**
*Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2330).*
- class **ProducerAckMarshaller**

*Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 2431).*

- class **ProducerIdMarshaller**

*Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 2465).*

- class **ProducerInfoMarshaller**

*Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 2486).*

- class **RemoveInfoMarshaller**

*Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 2556).*

- class **RemoveSubscriptionInfoMarshaller**

*Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 2580).*

- class **ReplayCommandMarshaller**

*Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 2599).*

- class **ResponseMarshaller**

*Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 2637).*

- class **SessionIdMarshaller**

*Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 2693).*

- class **SessionInfoMarshaller**

*Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 2713).*

- class **ShutdownInfoMarshaller**

*Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 2763).*

- class **SubscriptionInfoMarshaller**

*Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 2923).*

- class **TransactionIdMarshaller**

*Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3026).*

- class **TransactionInfoMarshaller**

*Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3052).*

- class **WireFormatInfoMarshaller**

*Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3174).*

- class **XATransactionIdMarshaller**

*Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3205).*

5.25 activemq::wireformat::openwire::marshal::v5 Namespace Reference

Data Structures

- class **ActiveMQBlobMessageMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 158).*
- class **ActiveMQBytesMessageMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 194).*
- class **ActiveMQDestinationMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 262).*
- class **ActiveMQMapMessageMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 296).*
- class **ActiveMQMessageMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 319).*
- class **ActiveMQObjectMessageMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 359).*
- class **ActiveMQQueueMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 394).*
- class **ActiveMQStreamMessageMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 447).*
- class **ActiveMQTempDestinationMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 470).*
- class **ActiveMQTempQueueMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 493).*
- class **ActiveMQTempTopicMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 517).*
- class **ActiveMQTextMessageMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 541).*
- class **ActiveMQTopicMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 565).*
- class **BaseCommandMarshaller**
*Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 623).*
- class **BrokerIdMarshaller**

*Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 705).*

- class **BrokerInfoMarshaller**

*Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 733).*

- class **ConnectionControlMarshaller**

*Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1071).*

- class **ConnectionErrorMarshaller**

*Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1095).*

- class **ConnectionIdMarshaller**

*Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1120).*

- class **ConnectionInfoMarshaller**

*Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1146).*

- class **ConsumerControlMarshaller**

*Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1182).*

- class **ConsumerIdMarshaller**

*Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1206).*

- class **ConsumerInfoMarshaller**

*Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1234).*

- class **ControlCommandMarshaller**

*Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1258).*

- class **DataArrayResponseMarshaller**

*Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1283).*

- class **DataResponseMarshaller**

*Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1318).*

- class **DestinationInfoMarshaller**

*Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1411).*

- class **DiscoveryEventMarshaller**

*Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1431).*

- class **ExceptionResponseMarshaller**

*Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1503).*

- class **FlushCommandMarshaller**

*Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 1591).*

- class **IntegerResponseMarshaller**

*Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 1685).*

- class **JournalQueueAckMarshaller**
*Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 1737).*
- class **JournalTopicAckMarshaller**
*Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 1762).*
- class **JournalTraceMarshaller**
*Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 1785).*
- class **JournalTransactionMarshaller**
*Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 1808).*
- class **KeepAliveInfoMarshaller**
*Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 1819).*
- class **LastPartialCommandMarshaller**
*Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 1858).*
- class **LocalTransactionIdMarshaller**
*Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 1894).*
- class **MarshallerFactory**
Used to create marshallers for a specific version of the wire protocol.
- class **MessageAckMarshaller**
*Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2064).*
- class **MessageDispatchMarshaller**
*Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2111).*
- class **MessageDispatchNotificationMarshaller**
*Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 2136).*
- class **MessageIdMarshaller**
*Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2157).*
- class **MessageMarshaller**
*Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2179).*
- class **MessagePullMarshaller**
*Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2219).*
- class **NetworkBridgeFilterMarshaller**
*Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2257).*
- class **PartialCommandMarshaller**
*Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2339).*
- class **ProducerAckMarshaller**

*Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 2435).*

- class **ProducerIdMarshaller**

*Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 2457).*

- class **ProducerInfoMarshaller**

*Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 2482).*

- class **RemoveInfoMarshaller**

*Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 2544).*

- class **RemoveSubscriptionInfoMarshaller**

*Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 2564).*

- class **ReplayCommandMarshaller**

*Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 2595).*

- class **ResponseMarshaller**

*Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 2628).*

- class **SessionIdMarshaller**

*Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 2681).*

- class **SessionInfoMarshaller**

*Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 2717).*

- class **ShutdownInfoMarshaller**

*Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 2775).*

- class **SubscriptionInfoMarshaller**

*Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 2919).*

- class **TransactionIdMarshaller**

*Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3029).*

- class **TransactionInfoMarshaller**

*Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3041).*

- class **WireFormatInfoMarshaller**

*Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3166).*

- class **XATransactionIdMarshaller**

*Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3213).*

5.26 activemq::wireformat::openwire::utils Namespace Reference

Data Structures

- class **BooleanStream**

Manages the writing and reading of boolean data streams to and from a data source such as a `DataInputStream` or `DataOutputStream`.

- class **HexTable**

*The **HexTable** (p. 1609) class maps hexadecimal strings to the value of an index into the table, i.e.*

- class **MessagePropertyInterceptor**

Used the base `ActiveMQMessage` class to intercept calls to get and set properties in order to capture the calls that use the reserved JMS properties and get and set them in the `OpenWire` Message properties.

- class **OpenwireStringSupport**

5.27 activemq::wireformat::stomp Namespace Reference

Data Structures

- class **StompCommandConstants**

- class **StompFrame**

A Stomp-level message frame that encloses all messages to and from the broker.

- class **StompHelper**

Utility Methods used when marshaling to and from `StompFrame`'s.

- class **StompWireFormat**

- class **StompWireFormatFactory**

Factory used to create the Stomp Wire Format instance.

5.28 cms Namespace Reference

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

Data Structures

- class **BytesMessage**

*A **BytesMessage** (p. 874) object is used to send a message containing a stream of unsigned bytes.*

- class **Closeable**

Interface for a class that implements the close method.

- class **CMSException**

CMS API Exception that is the base for all exceptions thrown from CMS classes.

- class **CMSProperties**

Interface for a Java-like properties object.

- class **CMSSecurityException**

This exception must be thrown when a provider rejects a user name/password submitted by a client.

- class **Connection**

The client's connection to its provider.

- class **ConnectionFactory**

*Defines the interface for a factory that creates connection objects, the **Connection** (p. 1052) objects returned implement the CMS **Connection** (p. 1052) interface and hide the CMS Provider specific implementation details behind that interface.*

- class **ConnectionMetaData**

*A **ConnectionMetaData** (p. 1154) object provides information describing the **Connection** (p. 1052) object.*

- class **DeliveryMode**

This is an Abstract class whose purpose is to provide a container for the delivery mode enumeration for CMS messages.

- class **Destination**

*A **Destination** (p. 1387) object encapsulates a provider-specific address.*

- class **ExceptionListener**

*If a CMS provider detects a serious problem, it notifies the client application through an **ExceptionListener** (p. 1483) that is registered with the **Connection** (p. 1052).*

- class **IllegalStateException**

This exception is thrown when a method is invoked at an illegal or inappropriate time or if the provider is not in an appropriate state for the requested operation.

- class **InvalidClientIdException**

This exception must be thrown when a client attempts to set a connection's client ID to a value that is rejected by a provider.

- class **InvalidDestinationException**

This exception must be thrown when a destination either is not understood by a provider or is no longer valid.

- class **InvalidSelectorException**

This exception must be thrown when a CMS client attempts to give a provider a message selector with invalid syntax.

- class **MapMessage**
*A **MapMessage** (p. 1982) object is used to send a set of name-value pairs.*
- class **Message**
Root of all messages.
- class **MessageConsumer**
*A client uses a **MessageConsumer** (p. 2080) to received messages from a destination.*
- class **MessageEOFException**
*This exception must be thrown when an unexpected end of stream has been reached when a **StreamMessage** (p. 2892) or **BytesMessage** (p. 874) is being read.*
- class **MessageFormatException**
This exception must be thrown when a CMS client attempts to use a data type not supported by a message or attempts to read data in a message as the wrong type.
- class **MessageListener**
*A **MessageListener** (p. 2165) object is used to receive asynchronously delivered messages.*
- class **MessageNotReadableException**
This exception must be thrown when a CMS client attempts to read a write-only message.
- class **MessageNotWriteableException**
This exception must be thrown when a CMS client attempts to write to a read-only message.
- class **MessageProducer**
*A client uses a **MessageProducer** (p. 2189) object to send messages to a **Destination** (p. 1387).*
- class **ObjectMessage**
Place holder for interaction with JMS systems that support Java, the C++ client is not responsible for deserializing the contained Object.
- class **Queue**
An interface encapsulating a provider-specific queue name.
- class **QueueBrowser**
*This class implements in interface for browsing the messages in a **Queue** (p. 2510) without removing them.*
- class **Session**
*A **Session** (p. 2663) object is a single-threaded context for producing and consuming messages.*
- class **Startable**
Interface for a class that implements the start method.
- class **Stoppable**
Interface for a class that implements the stop method.
- class **StreamMessage**

*Interface for a **StreamMessage** (p. 2892).*

- class **TemporaryQueue**

*Defines a Temporary **Queue** (p. 2510) based **Destination** (p. 1387).*

- class **TemporaryTopic**

*Defines a Temporary **Topic** (p. 3014) based **Destination** (p. 1387).*

- class **TextMessage**

Interface for a text message.

- class **Topic**

An interface encapsulating a provider-specific topic name.

- class **UnsupportedOperationException**

This exception must be thrown when a CMS client attempts use a CMS method that is not implemented or not supported by the CMS Provider in use.

5.28.1 Detailed Description

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements. See the NOTICE file distributed with this work for additional information regarding copyright ownership. The ASF licenses this file to You under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

5.29 decaf Namespace Reference

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

Namespaces

- namespace **internal**
- namespace **io**
- namespace **lang**
- namespace **net**
- namespace **nio**
- namespace **security**
- namespace **security_provider**
- namespace **util**

5.29.1 Detailed Description

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements. See the NOTICE file distributed with this work for additional information regarding copyright ownership. The ASF licenses this file to You under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

5.30 `decaf::internal` Namespace Reference

Namespaces

- namespace `io`
- namespace `net`
- namespace `nio`
- namespace `util`

Data Structures

- class `AprPool`
Wraps an APR pool object so that classes in `decaf` can create a static member for use in static methods where apr function calls that need a pool are made.
- class `DecafRuntime`
Handles APR initialization and termination.

5.31 `decaf::internal::io` Namespace Reference

Data Structures

- class `StandardErrorOutputStream`
Wrapper Around the Standard error Output facility on the current platform.
- class `StandardInputStream`
- class `StandardOutputStream`

5.32 `decaf::internal::net` Namespace Reference

Data Structures

- class `URIEncoderDecoder`

- class **URIHelper**

Helper class used by the URI classes in encoding and decoding of URI's.

- class **URIType**

Basic type object that holds data that composes a given URI.

5.33 decaf::internal::nio Namespace Reference

Data Structures

- class **BufferFactory**

*Factory class used by static methods in the **decaf::nio** (p. 109) package to create the various default version of the NIO interfaces.*

- class **ByteBuffer**

This class defines six categories of operations upon byte buffers:

- class **ByteArrayPerspective**

*This class extends **ByteArray** to create a reference counted byte array that can be held and used by several different **ByteBuffers** and allow them to know on destruction whose job it is to delete the perspective.*

- class **CharArrayBuffer**
- class **DoubleArrayBuffer**
- class **FloatArrayBuffer**
- class **IntArrayBuffer**
- class **LongArrayBuffer**
- class **ShortArrayBuffer**

5.34 decaf::internal::util Namespace Reference

Namespaces

- namespace **concurrent**

Data Structures

- class **ByteArrayAdapter**

This class adapts primitive type arrays to a base byte array so that the classes can inter-operate on the same base byte array without copying data.

- class **HexStringParser**
- class **TimerTaskHeap**

*A Binary Heap implemented specifically for the **Timer** class in **Decaf Util**.*

5.35 decaf::internal::util::concurrent Namespace Reference

Data Structures

- class **ConditionImpl**
- class **MutexImpl**
- class **SynchronizableImpl**

A convenience class used by some Decaf classes to implement the Synchronizable interface when there is no issues related to multiple inheritance.

- class **Transferer**

Shared internal API for dual stacks and queues.

- class **TransferQueue**

This extends Scherer-Scott dual queue algorithm, differing, among other ways, by using modes within nodes rather than marked pointers.

- class **TransferStack**

5.36 decaf::io Namespace Reference

Data Structures

- class **BlockingByteArrayInputStream**

This is a blocking version of a byte buffer stream.

- class **BufferedInputStream**

A wrapper around another input stream that performs a buffered read, where it reads more data than it needs in order to reduce the number of io operations on the input stream.

- class **BufferedOutputStream**

Wrapper around another output stream that buffers output before writing to the target output stream.

- class **ByteArrayInputStream**

*Simple implementation of **InputStream** (p.1630) that wraps around an STL Vector `std::vector<unsigned char>`.*

- class **ByteArrayOutputStream**

- class **Closeable**

Interface for a class that implements the close method.

- class **DataInputStream**

A data input stream lets an application read primitive Java data types from an underlying input stream in a machine-independent way.

- class **DataOutputStream**

A data output stream lets an application write primitive Java data types to an output stream in a portable way.

- class **EOFException**
- class **FilterInputStream**

A **FilterInputStream** (p. 1528) contains some other input stream, which it uses as its basic source of data, possibly transforming the data along the way or providing additional functionality.

- class **FilterOutputStream**

This class is the superclass of all classes that filter output streams.

- class **InputStream**

Base interface for an input stream.

- class **InterruptedIOException**
- class **IOException**
- class **OutputStream**

Base interface for an output stream.

- class **Reader**
- class **UnsupportedEncodingException**

Thrown when the the Character Encoding is not supported.

- class **UTFDataFormatException**

Thrown from classes that attempt to read or write a UTF-8 encoded string and an encoding error is encountered.

- class **Writer**

5.37 decaf::lang Namespace Reference

Namespaces

- namespace **exceptions**

Data Structures

- class **Appendable**
- class **Boolean**
- class **Byte**
- class **Character**
- class **CharSequence**

A **CharSequence** (p. 946) is a readable sequence of char values.

- class **Comparable**

This interface imposes a total ordering on the objects of each class that implements it.

- class **Double**
- class **Exception**
- class **Float**
- class **Integer**
- class **Iterable**

*Implementing this interface allows an object to be cast to an **Iterable** (p. 1715) type for generic collections API calls.*

- class **Long**

- class **Math**

*The class **Math** (p. 1999) contains methods for performing basic numeric operations such as the elementary exponential, logarithm, square root, and trigonometric functions.*

- class **Number**

*The abstract class **Number** (p. 2282) is the superclass of classes **Byte** (p. 776), **Double** (p. 1441), **Float** (p. 1544), **Integer** (p. 1652), **Long** (p. 1934), and **Short** (p. 2729).*

- class **AtomicRefCounter**

- struct **STATIC_CAST_TOKEN**

- struct **DYNAMIC_CAST_TOKEN**

- class **Pointer**

*Decaf's implementation of a Smart **Pointer** (p. 2343) that is a template on a Type and is Thread Safe if the default Reference Counter is used.*

- class **PointerComparator**

*This implementation of Comparator is designed to allows objects in a Collection to be sorted or tested for equality based on the value of the Object being Pointed to and not the value of the contained pointer in the **Pointer** (p. 2343) instance.*

- class **Runnable**

Interface for a runnable object - defines a task that can be run by a thread.

- class **Runtime**

- class **Short**

- class **System**

- class **ThreadGroup**

- class **Throwable**

This class represents an error that has occurred.

Functions

- template<typename T , typename R , typename U >
bool **operator**== (const **Pointer**< T, R > &left, const U *right)
- template<typename T , typename R , typename U >
bool **operator**== (const U *left, const **Pointer**< T, R > &right)
- template<typename T , typename R , typename U >
bool **operator**!= (const **Pointer**< T, R > &left, const U *right)
- template<typename T , typename R , typename U >
bool **operator**!= (const U *left, const **Pointer**< T, R > &right)

5.37.1 Function Documentation

5.37.1.1 `template<typename T , typename R , typename U > bool
decaf::lang::operator!= (const Pointer< T, R > & left, const U * right
) [inline]`

References `decaf::lang::Pointer< T, REFCOUNTER >::get()`.

5.37.1.2 `template<typename T , typename R , typename U > bool
decaf::lang::operator!= (const U * left, const Pointer< T, R > & right
) [inline]`

References `decaf::lang::Pointer< T, REFCOUNTER >::get()`.

5.37.1.3 `template<typename T , typename R , typename U > bool
decaf::lang::operator== (const Pointer< T, R > & left, const U *
right) [inline]`

References `decaf::lang::Pointer< T, REFCOUNTER >::get()`.

5.37.1.4 `template<typename T , typename R , typename U > bool
decaf::lang::operator== (const U * left, const Pointer< T, R > &
right) [inline]`

References `decaf::lang::Pointer< T, REFCOUNTER >::get()`.

5.38 decaf::lang::exceptions Namespace Reference

Data Structures

- class `ClassCastException`
- class `IllegalArgumentException`
- class `IllegalMonitorStateException`
- class `IllegalStateException`
- class `IllegalThreadStateException`
- class `IndexOutOfBoundsException`
- class `InterruptedException`
- class `InvalidStateException`
- class `NoSuchElementException`
- class `NullPointerException`
- class `NumberFormatException`
- class `RuntimeException`
- class `UnsupportedOperationException`

5.39 decaf::net Namespace Reference

Data Structures

- class `BindException`

- class **BufferedSocket**

*Buffered **Socket** (p. 2786) class that wraps a **Socket** (p. 2786) derived object and provides Buffered input and Output Streams to improve the efficiency of the reads and writes.*

- class **ConnectException**
- class **HttpRetryException**
- class **MalformedURLException**
- class **NoRouteToHostException**
- class **PortUnreachableException**
- class **ProtocolException**
- class **ServerSocket**

A server socket class (for testing purposes).

- class **Socket**
- class **SocketError**

Static utility class to simplify handling of error codes for socket operations.

- class **SocketException**

Exception for errors when manipulating sockets.

- class **SocketFactory**

***Socket** (p. 2786) Factory implementation for use in Creating Sockets.*

- class **SocketInputStream**

Input stream for performing reads on a socket.

- class **SocketOutputStream**

Output stream for performing write operations on a socket.

- class **SocketTimeoutException**
- class **TcpSocket**

Platform-independent implementation of the socket interface.

- class **UnknownHostException**
- class **UnknownServiceException**
- class **URI**

*This class represents an instance of a **URI** (p. 3096) as defined by RFC 2396.*

- class **URISyntaxException**
- class **URL**

*Class **URL** (p. 3133) represents a Uniform Resource Locator, a pointer to a "resource" on the World Wide Web.*

- class **URLDecoder**
- class **URLEncoder**

5.40 decaf::nio Namespace Reference

Data Structures

- class **Buffer**

A container for data of a specific primitive type.

- class **BufferOverflowException**
- class **BufferUnderflowException**
- class **ByteBuffer**

This class defines six categories of operations upon byte buffers:

- class **CharBuffer**

This class defines four categories of operations upon character buffers:

- class **DoubleBuffer**

This class defines four categories of operations upon double buffers:

- class **FloatBuffer**

This class defines four categories of operations upon float buffers:

- class **IntBuffer**

This class defines four categories of operations upon int buffers:

- class **InvalidMarkException**
- class **LongBuffer**

This class defines four categories of operations upon long long buffers:

- class **ReadOnlyBufferException**
- class **ShortBuffer**

This class defines four categories of operations upon short buffers:

5.41 decaf::security Namespace Reference

Namespaces

- namespace **auth**
- namespace **cert**

Data Structures

- class **GeneralSecurityException**
- class **InvalidKeyException**
- class **Key**

*The **Key** (p. 1835) interface is the top-level interface for all keys.*

- class **KeyException**
- class **NoSuchAlgorithmException**

- class **NoSuchProviderException**
- class **Principal**

Base interface for a principal, which can represent an individual or organization.

- class **PublicKey**
A public key.
- class **SignatureException**

5.42 decaf::security::auth Namespace Reference

Namespaces

- namespace **x500**

5.43 decaf::security::auth::x500 Namespace Reference

Data Structures

- class **X500Principal**

5.44 decaf::security::cert Namespace Reference

Data Structures

- class **Certificate**
Base interface for all identity certificates.
- class **CertificateEncodingException**
- class **CertificateException**
- class **CertificateExpiredException**
- class **CertificateNotYetValidException**
- class **CertificateParsingException**
- class **X509Certificate**
Base interface for all identity certificates.

5.45 decaf::security_provider Namespace Reference

Namespaces

- namespace **unix**

Data Structures

- class **SecurityProvider**
- class **SecurityProviderMap**
Lookup Map for Connector Factories.
- class **SecurityProviderRegistrar**
Registers the passed in provider into the provider map, this class can manage the lifetime of the registered provider (default behaviour).

5.46 decaf::security_provider::unix Namespace Reference

Namespaces

- namespace **openssl**

5.47 decaf::security_provider::unix::openssl Namespace Reference

Data Structures

- class **OpenSSLX500Principal**
The `OpenSSLX500Principal` (p. 2287) wraps around an `OpenSSL X509_NAME` structure.
- class **OpenSSLX509Certificate**

5.48 decaf::util Namespace Reference

Namespaces

- namespace **comparators**
- namespace **concurrent**
- namespace **logging**

Data Structures

- class **AbstractCollection**
This class provides a skeletal implementation of the `Collection` (p. 982) interface, to minimize the effort required to implement this interface.
- class **AbstractList**
This class provides a skeletal implementation of the `List` (p. 1865) interface to minimize the effort required to implement this interface backed by a "random access" data store (such as an array).
- class **AbstractMap**

*This class provides a skeletal implementation of the **Map** (p. 1970) interface, to minimize the effort required to implement this interface.*

- class **AbstractQueue**

*This class provides skeletal implementations of some **Queue** (p. 2507) operations.*

- class **AbstractSequentialList**

*This class provides a skeletal implementation of the **List** (p. 1865) interface to minimize the effort required to implement this interface backed by a "sequential access" data store (such as a linked list).*

- class **AbstractSet**

*This class provides a skeletal implementation of the **Set** (p. 2729) interface to minimize the effort required to implement this interface.*

- class **Collection**

The root interface in the collection hierarchy.

- class **Comparator**

A comparison function, which imposes a total ordering on some collection of objects.

- class **Date**

Wrapper class around a time value in milliseconds.

- class **Iterator**

Defines an object that can be used to iterate over the elements of a collection.

- class **List**

An ordered collection (also known as a sequence).

- class **ListIterator**

An iterator for lists that allows the programmer to traverse the list in either direction, modify the list during iteration, and obtain the iterator's current position in the list.

- class **Map**

***Map** (p. 1970) template that wraps around a `std::map` to provide a more user-friendly interface and to provide common functions that do not exist in `std::map`.*

- class **PriorityQueue**

An unbounded priority queue based on a binary heap algorithm.

- class **Properties**

Java-like properties class for mapping string names to string values.

- class **Queue**

A kind of collection provides advanced operations than other basic collections, such as insertion, extraction, and inspection.

- class **Random**

***Random** (p. 2513) Value Generator which is used to generate a stream of pseudorandom numbers.*

- class **Set**

A collection that contains no duplicate elements.

- class **StlList**

***List** (p. 1865) class that wraps the STL list object to provide a simpler interface and additional methods not provided by the STL type.*

- class **StlMap**

***Map** (p. 1970) template that wraps around a `std::map` to provide a more user-friendly interface and to provide common functions that do not exist in `std::map`.*

- class **StlQueue**

*The **Queue** (p. 2507) class accepts messages with an `psuh(m)` command where `m` is the message to be queued.*

- class **StlSet**

***Set** (p. 2729) template that wraps around a `std::set` to provide a more user-friendly interface and to provide common functions that do not exist in `std::set`.*

- class **StringTokenizer**

- class **Timer**

A facility for threads to schedule tasks for future execution in a background thread.

- class **TimerTask**

*A Base class for a task object that can be scheduled for one-time or repeated execution by a **Timer** (p. 2989).*

- class **UUID**

*A class that represents an immutable universally unique identifier (**UUID** (p. 3141)).*

5.49 decaf::util::comparators Namespace Reference

Data Structures

- class **Less**

*Simple **Less** (p. 1862) **Comparator** (p. 1011) that compares to elements to determine if the first is less than the second.*

5.50 decaf::util::concurrent Namespace Reference

Namespaces

- namespace **atomic**
- namespace **locks**

Data Structures

- class **ConditionHandle**
- class **MutexHandle**
- class **BrokenBarrierException**
- class **Callable**

A task that returns a result and may throw an exception.

- class **CancellationException**
- class **ConcurrentMap**

*Interface for a **Map** (p. 1970) type that provides additional atomic putIfAbsent, remove, and replace methods alongside the already available **Map** (p. 1970) interface.*

- class **ConcurrentStlMap**

***Map** (p. 1970) template that wraps around a std::map to provide a more user-friendly interface and to provide common functions that do not exist in std::map.*

- class **CountDownLatch**
- class **Delayed**

A mix-in style interface for marking objects that should be acted upon after a given delay.

- class **ExecutionException**
- class **Executor**

*An object that executes submitted **decaf.lang Runnable** (p. 2642) tasks.*

- class **ExecutorService**

*An **Executor** (p. 1509) that provides methods to manage termination and methods that can produce a **Future** (p. 1597) for tracking progress of one or more asynchronous tasks.*

- class **Future**

*A **Future** (p. 1597) represents the result of an asynchronous computation.*

- class **Lock**

A wrapper class around a given synchronization mechanism that provides automatic release upon destruction.

- class **Mutex**

***Mutex** (p. 2239) object that offers recursive support on all platforms as well as providing the ability to use the standard wait / notify pattern used in languages like Java.*

- class **PooledThread**
- class **PooledThreadListener**

*Abstract Listener Interface for users of **ThreadPool** (p. 2977).*

- class **RejectedExecutionException**
- class **RejectedExecutionHandler**

*A handler for tasks that cannot be executed by a **ThreadPoolExecutor** (p. ??).*

- class **Semaphore**

A counting semaphore.

- class **Synchronizable**

The interface for all synchronizable objects (that is, objects that can be locked and unlocked).

- class **SynchronousQueue**

A `BlockingQueue` blocking queue} in which each insert operation must wait for a corresponding remove operation by another thread, and vice versa.

- class **TaskListener**

- class **ThreadFactory**

*public interface **ThreadFactory** (p. 2976)*

- class **ThreadPool**

Defines a Thread Pool object that implements the functionality of pooling threads to perform user tasks.

- class **TimeoutException**

- class **TimeUnit**

*A **TimeUnit** (p. 3005) represents time durations at a given unit of granularity and provides utility methods to convert across units, and to perform timing and delay operations in these units.*

5.51 decaf::util::concurrent::atomic Namespace Reference

Data Structures

- class **AtomicBoolean**

A boolean value that may be updated atomically.

- class **AtomicInteger**

An int value that may be updated atomically.

- class **AtomicReference**

An Pointer reference that may be updated atomically.

5.52 decaf::util::concurrent::locks Namespace Reference

Data Structures

- class **Condition**

***Condition** (p. 1040) factors out the **Mutex** (p. 2239) monitor methods (wait, notify and notifyAll) into distinct objects to give the effect of having multiple wait-sets per object, by combining them with the use of arbitrary **Lock** (p. 1898) implementations.*

- class **Lock**

***Lock** (p. 1898) implementations provide more extensive locking operations than can be obtained using synchronized statements.*

- class **LockSupport**

Basic thread blocking primitives for creating locks and other synchronization classes.

- class **ReadWriteLock**

*A **ReadWriteLock** (p. 2522) maintains a pair of associated locks, one for read-only operations and one for writing.*

- class **ReentrantLock**

*A reentrant mutual exclusion **Lock** (p. 1898) with extended capabilities.*

5.53 decaf::util::logging Namespace Reference

Data Structures

- class **Filter**

*A **Filter** (p. 1527) can be used to provide fine grain control over what is logged, beyond the control provided by log levels.*

- class **Formatter**

*A **Formatter** (p. 1595) provides support for formatting **LogRecords**.*

- class **Handler**

*A **Handler** (p. 1604) object takes log messages from a **Logger** (p. 1908) and exports them.*

- class **Logger**

- class **LoggerHierarchy**

- class **LogManager**

*There is a single global **LogManager** (p. 1923) object that is used to maintain a set of shared state about **Loggers** and log services.*

- class **LogRecord**

- class **LogWriter**

- class **MarkBlockLogger**

Defines a class that can be used to mark the entry and exit from scoped blocks.

- class **PropertiesChangeListener**

*Defines the interface that classes can use to listen for change events on **Properties** (p. 2494).*

- class **SimpleFormatter**

*Print a brief summary of the **LogRecord** (p. 1928) in a human readable format.*

- class **SimpleLogger**

- class **StreamHandler**

Enumerations

- enum **Level** {
 Off, **Null**, **Markblock**, **Debug**,
 Info, **Warn**, **Error**, **Fatal**,
 Throwing }
 Defines an enumeration for logging levels.

5.53.1 Enumeration Type Documentation

5.53.1.1 enum decaf::util::logging::Level

Defines an enumeration for logging levels.

Enumerator:

Off
Null
Markblock
Debug
Info
Warn
Error
Fatal
Throwing

5.54 std Namespace Reference

Data Structures

- struct **less**< **decaf::lang::Pointer**< **T** > >
 An override of the less function object so that the Pointer objects can be stored in STL Maps, etc.

Chapter 6

Data Structure Documentation

6.1 decaf::util::AbstractCollection< E > Class Template Reference

This class provides a skeletal implementation of the **Collection** (p. 982) interface, to minimize the effort required to implement this interface.

```
#include <src/main/decaf/util/AbstractCollection.h>
```

Inheritance diagram for decaf::util::AbstractCollection< E >:

Public Member Functions

- virtual **~AbstractCollection** ()
- **AbstractCollection**< E > & **operator=** (const **AbstractCollection**< E > &collection)
Assignment Operator, copy element from the source collection to this collection after clearing any element stored in this collection.
- virtual bool **add** (const E &value DECAF_UNUSED)
throw (lang::exceptions::UnsupportedOperationException,
lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException)
Ensures that this collection contains the specified element (optional operation).
- virtual bool **addAll** (const **Collection**< E > &collection)
throw (lang::exceptions::UnsupportedOperationException,
lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException)
Adds all of the elements in the specified collection to this collection (optional operation).
- virtual void **clear** () throw (lang::exceptions::UnsupportedOperationException)
Removes all of the elements from this collection (optional operation).
- virtual void **copy** (const **Collection**< E > &collection)
*Renders this **Collection** (p. 982) as a Copy of the given **Collection** (p. 982).*

- virtual bool **contains** (const E &value) const throw (lang::Exception)
Returns true if this collection contains the specified element.
- virtual bool **containsAll** (const **Collection**< E > &collection) const throw (lang::Exception)
Returns true if this collection contains all of the elements in the specified collection.
- virtual bool **equals** (const **Collection**< E > &collection) const
*Answers true if this **Collection** (p. 982) and the one given are the same size and if each element contained in the **Collection** (p. 982) given is equal to an element contained in this collection.*
- virtual bool **isEmpty** () const
Returns true if this collection contains no elements.
- virtual bool **remove** (const E &value) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException)
Removes a single instance of the specified element from this collection, if it is present (optional operation).
- virtual bool **removeAll** (const **Collection**< E > &collection) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException)
Removes all of this collection's elements that are also contained in the specified collection (optional operation).
- virtual bool **retainAll** (const **Collection**< E > &collection) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException)
Retains only the elements in this collection that are contained in the specified collection (optional operation).
- virtual std::vector< E > **toArray** () const
*Answers an STL vector containing copies of all elements contained in this **Collection** (p. 982).*
- virtual void **lock** () throw (decaf::lang::exceptions::RuntimeException)
Locks the object.
- virtual bool **tryLock** () throw (decaf::lang::exceptions::RuntimeException)
Attempts to Lock the object, if the lock is already held by another thread than this method returns false.
- virtual void **unlock** () throw (decaf::lang::exceptions::RuntimeException)
Unlocks the object.
- virtual void **wait** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (long long millisecs) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)

Waits on a signal from this object, which is generated by a call to Notify.

- virtual void **wait** (long long millisecs, int nanos) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)

Waits on a signal from this object, which is generated by a call to Notify.

- virtual void **notify** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)

Signals a waiter on this object that it can now wake up and continue.

- virtual void **notifyAll** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)

Signals the waiters on this object that it can now wake up and continue.

Protected Attributes

- util::concurrent::Mutex mutex

6.1.1 Detailed Description

template<typename E> class decaf::util::AbstractCollection< E >

This class provides a skeletal implementation of the **Collection** (p. 982) interface, to minimize the effort required to implement this interface. To implement an unmodifiable collection, the programmer needs only to extend this class and provide implementations for the iterator and size methods. (The iterator returned by the iterator method must implement hasNext and next.)

To implement a modifiable collection, the programmer must additionally override this class's add method (which otherwise throws an UnsupportedOperationException), and the iterator returned by the iterator method must additionally implement its remove method.

The programmer should generally provide a void (no argument) and **Collection** (p. 982) constructor, as per the recommendation in the **Collection** (p. 982) interface specification.

The documentation for each non-abstract method in this class describes its implementation in detail. Each of these methods may be overridden if the collection being implemented admits a more efficient implementation.

Since

1.0

6.1.2 Constructor & Destructor Documentation

6.1.2.1 `template<typename E> virtual decaf::util::AbstractCollection< E >::~~AbstractCollection () [inline, virtual]`

6.1.3 Member Function Documentation

6.1.3.1 `template<typename E> virtual bool decaf::util::AbstractCollection< E >::add (const E &value DECAF_UNUSED) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException) [inline, virtual]`

Ensures that this collection contains the specified element (optional operation).

Returns true if this collection changed as a result of the call. (Returns false if this collection does not permit duplicates and already contains the specified element.)

Collections that support this operation may place limitations on what elements may be added to this collection. **Collection** (p.982) classes should clearly specify in their documentation any restrictions on what elements may be added.

If a collection refuses to add a particular element for any reason other than that it already contains the element, it must throw an exception (rather than returning false). This preserves the invariant that a collection always contains the specified element after this call returns.

This implementation always throws an `UnsupportedOperationException`.

Parameters

value - The element that must be ensured to be in this collection.

Returns

true if the collection was changed as a result of this call.

Exceptions

UnsupportedOperationException if the add operation is not supported by this collection

IllegalArgumentException if some property of the element prevents it from being added to this collection

IllegalStateException if the element cannot be added at this time due to insertion restrictions

Referenced by `decaf::util::AbstractCollection< Pointer< BackupTransport > >::addAll()`, `decaf::util::AbstractCollection< Pointer< BackupTransport > >::copy()`, and `decaf::util::AbstractCollection< Pointer< BackupTransport > >::operator=()`.

6.1.3.2 `template<typename E> virtual bool decaf::util::AbstractCollection< E >::addAll (const Collection< E > & collection) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException) [inline, virtual]`

Adds all of the elements in the specified collection to this collection (optional operation).

The behavior of this operation is undefined if the specified collection is modified while the operation is in progress. (This implies that the behavior of this call is undefined if the specified collection is this collection, and this collection is nonempty.)

This implementation iterates over the specified collection, and adds each object returned by the iterator to this collection, in turn.

Note that this implementation will throw an `UnsupportedOperationException` unless `add` is overridden (assuming the specified collection is non-empty).

Parameters

collection - The **Collection** (p.982) whose elements are to be added to this **Collection** (p.982).

Returns

true if the collection was changed as a result of this call.

Exceptions

UnsupportedOperationException if the `addAll` operation is not supported by this collection

IllegalArgumentException if some property of the element prevents it from being added to this collection

IllegalStateException if the element cannot be added at this time due to insertion restrictions

Implements **decaf::util::Collection< E >** (p.985).

Reimplemented in **decaf::util::AbstractQueue< E >** (p.135).

```
6.1.3.3  template<typename E> virtual void decaf::util::AbstractCollection< E
>::clear (    ) throw ( lang::exceptions::UnsupportedOperationException )
[inline, virtual]
```

Removes all of the elements from this collection (optional operation).

The collection will be empty after this method returns.

This implementation iterates over this collection, removing each element using the **Iterator.remove** (p.1718) operation. Most implementations will probably choose to override this method for efficiency.

Note that this implementation will throw an `UnsupportedOperationException` if the iterator returned by this collection's iterator method does not implement the `remove` method and this collection is non-empty.

Exceptions

UnsupportedOperationException if the `clear` operation is not supported by this collection

Implements **decaf::util::Collection< E >** (p.985).

Reimplemented in **decaf::util::AbstractQueue< E >** (p.136), **decaf::util::PriorityQueue< E >** (p.2417), **decaf::util::StlList< E >** (p.2840), **decaf::util::StlSet< E >** (p.2869), **decaf::util::StlList< CompositeTask * >** (p.2840), **decaf::util::StlList< URI >** (p.2840),

decaf::util::StlList< Pointer< DestinationInfo > > (p. 2840), **decaf::util::StlList< PrimitiveValueNode >** (p. 2840), **decaf::util::StlList< Pointer< Command > >** (p. 2840), **decaf::util::StlList< Pointer< BackupTransport > >** (p. 2840), **decaf::util::StlSet< transport::TransportListener * >** (p. 2869), **decaf::util::StlSet< Pointer< Synchronization > >** (p. 2869), and **decaf::util::StlSet< ActiveMQSession * >** (p. 2869).

Referenced by **decaf::util::AbstractCollection< Pointer< BackupTransport > >::copy()**, and **decaf::util::AbstractCollection< Pointer< BackupTransport > >::operator=()**.

6.1.3.4 `template<typename E> virtual bool decaf::util::AbstractCollection< E >::contains (const E & value) const throw (lang::Exception) [inline, virtual]`

Returns true if this collection contains the specified element.

This implementation iterates over the elements in the collection, checking each element in turn for equality with the specified element.

Parameters

value - the value whose presence is to be queried for in this **Collection** (p. 982).

Returns

true if the value is contained in this collection

Exceptions

Exception if an error occurs,

Implements **decaf::util::Collection< E >** (p. 986).

Reimplemented in **decaf::util::StlList< E >** (p. 2840), **decaf::util::StlSet< E >** (p. 2869), **decaf::util::StlList< CompositeTask * >** (p. 2840), **decaf::util::StlList< URI >** (p. 2840), **decaf::util::StlList< Pointer< DestinationInfo > >** (p. 2840), **decaf::util::StlList< PrimitiveValueNode >** (p. 2840), **decaf::util::StlList< Pointer< Command > >** (p. 2840), **decaf::util::StlList< Pointer< BackupTransport > >** (p. 2840), **decaf::util::StlSet< transport::TransportListener * >** (p. 2869), **decaf::util::StlSet< Pointer< Synchronization > >** (p. 2869), and **decaf::util::StlSet< ActiveMQSession * >** (p. 2869).

Referenced by **decaf::util::AbstractCollection< Pointer< BackupTransport > >::containsAll()**.

6.1.3.5 `template<typename E> virtual bool decaf::util::AbstractCollection< E >::containsAll (const Collection< E > & collection) const throw (lang::Exception) [inline, virtual]`

Returns true if this collection contains all of the elements in the specified collection.

This implementation iterates over the specified collection, checking each element returned by the iterator in turn to see if it's contained in this collection. If all elements are so contained true is returned, otherwise false.

Parameters

collection collection to be checked for containment in this collection

Returns

true if this collection contains all of the elements in the specified collection.

Exceptions

Exception if an error occurs,

Implements **decaf::util::Collection< E >** (p. 986).

Referenced by **decaf::util::AbstractCollection< Pointer< BackupTransport > >::equals()**.

6.1.3.6 **template<typename E> virtual void decaf::util::AbstractCollection< E >::copy (const Collection< E > & collection) [inline, virtual]**

Renders this **Collection** (p. 982) as a Copy of the given **Collection** (p. 982).

This implementation iterates over the contents of the given collection adding each to this collection after first calling this **Collection**'s clear method.

Parameters

collection - the collection to mirror.

6.1.3.7 **template<typename E> virtual bool decaf::util::AbstractCollection< E >::equals (const Collection< E > & collection) const [inline, virtual]**

Answers true if this **Collection** (p. 982) and the one given are the same size and if each element contained in the **Collection** (p. 982) given is equal to an element contained in this collection.

Parameters

collection - The **Collection** (p. 982) to be compared to this one.

Returns

true if this **Collection** (p. 982) is equal to the one given.

Implements **decaf::util::Collection< E >** (p. 987).

6.1.3.8 **template<typename E> virtual bool decaf::util::AbstractCollection< E >::isEmpty () const [inline, virtual]**

Returns true if this collection contains no elements.

This implementation returns **size()** (p. 990) == 0.

Returns

true if the size method return 0.

Implements **decaf::util::Collection< E >** (p. 987).

Reimplemented in **decaf::util::StlList< E >** (p. 2841), **decaf::util::StlSet< E >** (p. 2870), **decaf::util::StlList< CompositeTask * >** (p. 2841), **decaf::util::StlList< URI >** (p. 2841),

`decaf::util::StlList< Pointer< DestinationInfo > >` (p. 2841), `decaf::util::StlList< PrimitiveValueNode >` (p. 2841), `decaf::util::StlList< Pointer< Command > >` (p. 2841), `decaf::util::StlList< Pointer< BackupTransport > >` (p. 2841), `decaf::util::StlSet< transport::TransportListener * >` (p. 2870), `decaf::util::StlSet< Pointer< Synchronization > >` (p. 2870), and `decaf::util::StlSet< ActiveMQSession * >` (p. 2870).

Referenced by `decaf::util::AbstractQueue< E >::clear()`.

6.1.3.9 `template<typename E> virtual void decaf::util::AbstractCollection< E >::lock () throw (decaf::lang::exceptions::RuntimeException) [inline, virtual]`

Locks the object.

Exceptions

RuntimeException if an error occurs while locking the object.

Implements `decaf::util::concurrent::Synchronizable` (p. 2932).

6.1.3.10 `template<typename E> virtual void decaf::util::AbstractCollection< E >::notify () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException) [inline, virtual]`

Signals a waiter on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

RuntimeException if an error occurs while notifying one of the waiting threads.

Implements `decaf::util::concurrent::Synchronizable` (p. 2933).

6.1.3.11 `template<typename E> virtual void decaf::util::AbstractCollection< E >::notifyAll () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException) [inline, virtual]`

Signals the waiters on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

RuntimeException if an error occurs while notifying the waiting threads.

Implements `decaf::util::concurrent::Synchronizable` (p. 2934).

6.1.3.12 `template<typename E> AbstractCollection<E>&
decaf::util::AbstractCollection< E >::operator= (const
AbstractCollection< E > & collection) [inline]`

Assignment Operator, copy element from the source collection to this collection after clearing any element stored in this collection.

Parameters

collection - the collection to copy

Returns

a reference to this collection

6.1.3.13 `template<typename E> virtual bool decaf::util::AbstractCollection<
E >::remove (const E & value) throw (
lang::exceptions::UnsupportedOperationException,
lang::exceptions::IllegalArgumentException) [inline, virtual]`

Removes a single instance of the specified element from this collection, if it is present (optional operation).

More formally, removes the first element *e* such that *e* == *o*, if this collection contains one or more such elements. Returns true if this collection contained the specified element (or equivalently, if this collection changed as a result of the call).

This implementation iterates over the collection looking for the specified element. If it finds the element, it removes the element from the collection using the iterator's remove method.

Note that this implementation throws an UnsupportedOperationException if the iterator returned by this collection's iterator method does not implement the remove method and this collection contains the specified object.

Parameters

value - element to be removed from this collection, if present

Returns

true if an element was removed as a result of this call

Exceptions

UnsupportedOperationException if the remove operation is not supported by this collection.

IllegalArgumentException If the value is not a valid entry for this **Collection** (p. 982).

Implements **decaf::util::Collection< E >** (p. 988).

Reimplemented in **decaf::util::PriorityQueue< E >** (p. 2419), **decaf::util::StlList< E >** (p. 2843), **decaf::util::StlSet< E >** (p. 2870), **decaf::util::StlList< CompositeTask * >** (p. 2843), **decaf::util::StlList< URI >** (p. 2843), **decaf::util::StlList< Pointer< DestinationInfo > >** (p. 2843), **decaf::util::StlList< PrimitiveValueNode >** (p. 2843), **decaf::util::StlList< Pointer< Command > >** (p. 2843), **decaf::util::StlList< Pointer< BackupTransport > >** (p. 2843), **decaf::util::StlSet< transport::TransportListener * >** (p. 2870), **decaf::util::StlSet< Pointer< Synchronization > >** (p. 2870), and **decaf::util::StlSet< ActiveMQSession * >** (p. 2870).

```

6.1.3.14  template<typename E> virtual bool decaf::util::AbstractCollection<
            E >::removeAll (  const Collection< E > &  collection  )
            throw ( lang::exceptions::UnsupportedOperationException,
                    lang::exceptions::IllegalArgumentException ) [inline, virtual]

```

Removes all of this collection's elements that are also contained in the specified collection (optional operation).

After this call returns, this collection will contain no elements in common with the specified collection.

This implementation iterates over this collection, checking each element returned by the iterator in turn to see if it's contained in the specified collection. If it's so contained, it's removed from this collection with the iterator's remove method.

Note that this implementation will throw an `UnsupportedOperationException` if the iterator returned by the iterator method does not implement the remove method and this collection contains one or more elements in common with the specified collection.

Parameters

collection - collection containing elements to be removed from this collection

Returns

true if this collection changed as a result of the call

Exceptions

UnsupportedOperationException if the remove operation is not supported by this collection

IllegalArgumentException.

Implements `decaf::util::Collection< E >` (p. 988).

Reimplemented in `decaf::util::AbstractSet< E >` (p. 139), `decaf::util::AbstractSet< transport::TransportListener * >` (p. 139), `decaf::util::AbstractSet< Pointer< Synchronization > >` (p. 139), and `decaf::util::AbstractSet< ActiveMQSession * >` (p. 139).

```

6.1.3.15  template<typename E> virtual bool decaf::util::AbstractCollection<
            E >::retainAll (  const Collection< E > &  collection  )
            throw ( lang::exceptions::UnsupportedOperationException,
                    lang::exceptions::IllegalArgumentException ) [inline, virtual]

```

Retains only the elements in this collection that are contained in the specified collection (optional operation).

In other words, removes from this collection all of its elements that are not contained in the specified collection.

This implementation iterates over this collection, checking each element returned by the iterator in turn to see if it's contained in the specified collection. If it's not so contained, it's removed from this collection with the iterator's remove method.

Note that this implementation will throw an `UnsupportedOperationException` if the iterator returned by the iterator method does not implement the remove method and this collection contains one or more elements not present in the specified collection.

Parameters

collection - collection containing elements to be retained in this collection

Returns

true if this collection changed as a result of the call

Exceptions

UnsupportedOperationException if the remove operation is not supported by this collection

IllegalArgumentException.

Implements **decaf::util::Collection< E >** (p. 989).

6.1.3.16 `template<typename E> virtual std::vector<E>
decaf::util::AbstractCollection< E >::toArray () const [inline,
virtual]`

Answers an STL vector containing copies of all elements contained in this **Collection** (p. 982).

All the elements in the array will not be referenced by the collection. The elements in the returned array will be sorted to the same order as those returned by the iterator of this collection itself if the collection guarantees the order.

Returns

an vector of copies of all the elements from this **Collection** (p. 982)

Implements **decaf::util::Collection< E >** (p. 990).

6.1.3.17 `template<typename E> virtual bool decaf::util::AbstractCollection< E
>::tryLock () throw (decaf::lang::exceptions::RuntimeException)
[inline, virtual]`

Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

Returns

true if the lock was acquired, false if it is already held by another thread.

Exceptions

RuntimeException if an error occurs while locking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2935).

6.1.3.18 `template<typename E> virtual void decaf::util::AbstractCollection< E
>::unlock () throw (decaf::lang::exceptions::RuntimeException)
[inline, virtual]`

Unlocks the object.

Exceptions

RuntimeException if an error occurs while unlocking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2936).

```
6.1.3.19  template<typename E> virtual void decaf::util::AbstractCollection<
           E >::wait (      ) throw ( decaf::lang::exceptions::RuntimeException,
           decaf::lang::exceptions::IllegalMonitorStateException,
           decaf::lang::exceptions::InterruptedException ) [inline, virtual]
```

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling.

Exceptions

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2937).

```
6.1.3.20  template<typename E> virtual void decaf::util::AbstractCollection<
           E >::wait ( long long millisecs, int nanos )
           throw ( decaf::lang::exceptions::RuntimeException,
           decaf::lang::exceptions::IllegalArgumentOutOfRangeException,
           decaf::lang::exceptions::IllegalMonitorStateException,
           decaf::lang::exceptions::InterruptedException ) [inline, virtual]
```

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

Parameters

millisecs the time in milliseconds to wait, or WAIT_INFINITE

nanos additional time in nanoseconds with a range of 0-999999

Exceptions

IllegalArgumentOutOfRangeException if an error occurs or the nanos argument is not in the range of [0-999999]

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2940).

6.1.3.21 `template<typename E> virtual void decaf::util::AbstractCollection< E >::wait (long long millisecs) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval.

Parameters

millisecs the time in milliseconds to wait, or WAIT_INFINITE

Exceptions

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements `decaf::util::concurrent::Synchronizable` (p. 2938).

6.1.4 Field Documentation

6.1.4.1 `template<typename E> util::concurrent::Mutex decaf::util::AbstractCollection< E >::mutex [mutable, protected]`

Referenced by `decaf::util::AbstractCollection< Pointer< BackupTransport > >::lock()`, `decaf::util::AbstractCollection< Pointer< BackupTransport > >::notify()`, `decaf::util::AbstractCollection< Pointer< BackupTransport > >::notifyAll()`, `decaf::util::AbstractCollection< Pointer< BackupTransport > >::tryLock()`, `decaf::util::AbstractCollection< Pointer< BackupTransport > >::unlock()`, and `decaf::util::AbstractCollection< Pointer< BackupTransport > >::wait()`.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/AbstractCollection.h`

6.2 decaf::util::AbstractList< E > Class Template Reference

This class provides a skeletal implementation of the **List** (p. 1865) interface to minimize the effort required to implement this interface backed by a "random access" data store (such as an array).

`#include <src/main/decaf/util/AbstractList.h>`

Inheritance diagram for `decaf::util::AbstractList< E >`:

Public Member Functions

- `virtual ~AbstractList ()`

6.2.1 Detailed Description

template<typename E> class decaf::util::AbstractList< E >

This class provides a skeletal implementation of the **List** (p.1865) interface to minimize the effort required to implement this interface backed by a "random access" data store (such as an array). For sequential access data (such as a linked list), **AbstractSequentialList** (p.137) should be used in preference to this class.

To implement an unmodifiable list, the programmer needs only to extend this class and provide implementations for the `get(int)` and `size()` (p.990) methods.

To implement a modifiable list, the programmer must additionally override the `set(int, E)` method (which otherwise throws an `UnsupportedOperationException`). If the list is variable-size the programmer must additionally override the `add(int, E)` and `remove(int)` methods.

The programmer should generally provide a void (no argument) and collection constructor, as per the recommendation in the **Collection** (p.982) interface specification.

Unlike the other abstract collection implementations, the programmer does not have to provide an iterator implementation; the iterator and list iterator are implemented by this class, on top of the "random access" methods: `get(int)`, `set(int, E)`, `add(int, E)` and `remove(int)`.

The documentation for each non-abstract method in this class describes its implementation in detail. Each of these methods may be overridden if the collection being implemented admits a more efficient implementation.

Since

1.0

6.2.2 Constructor & Destructor Documentation

6.2.2.1 template<typename E > virtual decaf::util::AbstractList< E >::~~AbstractList () [inline, virtual]

The documentation for this class was generated from the following file:

- `src/main/decaf/util/AbstractList.h`

6.3 decaf::util::AbstractMap< K, V, COMPARATOR > Class Template Reference

This class provides a skeletal implementation of the **Map** (p.1970) interface, to minimize the effort required to implement this interface.

`#include <src/main/decaf/util/AbstractMap.h>`

Inheritance diagram for `decaf::util::AbstractMap< K, V, COMPARATOR >`:

Public Member Functions

- `virtual ~AbstractMap ()`

6.3.1 Detailed Description

```
template<typename K, typename V, typename COMPARATOR> class
decaf::util::AbstractMap< K, V, COMPARATOR >
```

This class provides a skeletal implementation of the **Map** (p. 1970) interface, to minimize the effort required to implement this interface. To implement an unmodifiable map, the programmer needs only to extend this class and provide an implementation for the `entrySet` method, which returns a set-view of the map's mappings. Typically, the returned set will, in turn, be implemented atop **AbstractSet** (p. 138). This set should not support the `add` or `remove` methods, and its iterator should not support the `remove` method.

To implement a modifiable map, the programmer must additionally override this class's `put` method (which otherwise throws an `UnsupportedOperationException`), and the iterator returned by `entrySet().iterator()` must additionally implement its `remove` method.

The programmer should generally provide a void (no argument) and map constructor, as per the recommendation in the **Map** (p. 1970) interface specification.

The documentation for each non-abstract method in this class describes its implementation in detail. Each of these methods may be overridden if the map being implemented admits a more efficient implementation.

Since

1.0

6.3.2 Constructor & Destructor Documentation

```
6.3.2.1 template<typename K , typename V , typename COMPARATOR >
virtual decaf::util::AbstractMap< K, V, COMPARATOR >::~~AbstractMap
( ) [inline, virtual]
```

The documentation for this class was generated from the following file:

- `src/main/decaf/util/AbstractMap.h`

6.4 decaf::util::AbstractQueue< E > Class Template Reference

This class provides skeletal implementations of some **Queue** (p. 2507) operations.

```
#include <src/main/decaf/util/AbstractQueue.h>
```

Inheritance diagram for `decaf::util::AbstractQueue< E >`:

Public Member Functions

- `AbstractQueue()`
- `virtual ~AbstractQueue()`

- virtual bool **add** (const E &value) throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException)

Inserts the specified element into this queue if it is possible to do so immediately without violating capacity restrictions, returning true upon success and throwing an IllegalStateException if no space is currently available.

- virtual bool **addAll** (const Collection< E > &collection) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException)

Adds all the elements of a collection to the queue.

- virtual E **remove** () throw (decaf::lang::exceptions::NoSuchElementException)

Retrieves and removes the head of this queue.

- virtual E **element** () const throw (decaf::lang::exceptions::NoSuchElementException)

Retrieves, but does not remove, the head of this queue.

- virtual void **clear** () throw (lang::exceptions::UnsupportedOperationException)

Removes all elements of the queue.

6.4.1 Detailed Description

template<typename E> class decaf::util::AbstractQueue< E >

This class provides skeletal implementations of some **Queue** (p.2507) operations. Methods add, remove, and element are based on offer, poll, and peek, respectively.

A **Queue** (p. 2507) implementation that extends this class must minimally define a method **Queue** (p. 2507). **offer(E)** which does not permit insertion of null elements, along with methods **Queue** (p. 2507). **peek()** (p. 2509), **Queue.poll()** (p. 2509), **Collection.size()** (p. 990), and a **Collection.iterator()** (p. 1716) supporting **Iterator.remove()** (p. 1718). Typically, additional methods will be overridden as well. If these requirements cannot be met, consider instead subclassing **AbstractCollection** (p. 119).

Since

1.0

6.4.2 Constructor & Destructor Documentation

6.4.2.1 `template<typename E > decaf::util::AbstractQueue< E >::AbstractQueue () [inline]`

6.4.2.2 `template<typename E > virtual decaf::util::AbstractQueue< E >::~~AbstractQueue () [inline, virtual]`

6.4.3 Member Function Documentation

6.4.3.1 `template<typename E > virtual bool decaf::util::AbstractQueue< E >::add (const E & value) throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException) [inline, virtual]`

Inserts the specified element into this queue if it is possible to do so immediately without violating capacity restrictions, returning true upon success and throwing an `IllegalStateException` if no space is currently available.

This implementation returns true if offer succeeds, else throws an `IllegalStateException`.

Parameters

value - the element to offer to the **Queue** (p. 2507).

Returns

true if the add succeeds.

Exceptions

IllegalArgumentException if the element cannot be added.

Implements `decaf::util::Collection< E >` (p. 984).

Reimplemented in `decaf::util::PriorityQueue< E >` (p. 2416).

References `decaf::util::Queue< E >::offer()`.

6.4.3.2 `template<typename E > virtual bool decaf::util::AbstractQueue< E >::addAll (const Collection< E > & collection) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException) [inline, virtual]`

Adds all the elements of a collection to the queue.

If the collection is the queue itself, then an `IllegalArgumentException` will be thrown out. If during the process, some runtime exception is thrown out, then part of the elements in the collection that have successfully added will remain in the queue.

The result of the method is undefined if the collection is modified during the process of the method.

Parameters

collection - the collection to be added to the queue.

Returns

true if the operation succeeds.

Exceptions

IllegalArgumentException If the collection to be added to the queue is the queue itself.

Reimplemented from **decaf::util::AbstractCollection**< **E** > (p. 122).

6.4.3.3 `template<typename E> virtual void decaf::util::AbstractQueue< E >::clear () throw (lang::exceptions::UnsupportedOperationException) [inline, virtual]`

Removes all elements of the queue.

This implementation repeatedly invokes poll until it returns the empty marker.

Reimplemented from **decaf::util::AbstractCollection**< **E** > (p. 123).

Reimplemented in **decaf::util::PriorityQueue**< **E** > (p. 2417).

References **decaf::util::AbstractCollection**< **E** >::isEmpty(), and **decaf::util::Queue**< **E** >::poll().

6.4.3.4 `template<typename E> virtual E decaf::util::AbstractQueue< E >::element () const throw (decaf::lang::exceptions::NoSuchElementException) [inline, virtual]`

Retrieves, but does not remove, the head of this queue.

This method differs from peek only in that it throws an exception if this queue is empty.

This implementation returns the result of peek unless the queue is empty.

Returns

the element in the head of the queue.

Exceptions

NoSuchElementException if the queue is empty.

Implements **decaf::util::Queue**< **E** > (p. 2508).

References **decaf::util::Queue**< **E** >::peek().

6.4.3.5 `template<typename E> virtual E decaf::util::AbstractQueue< E >::remove () throw (decaf::lang::exceptions::NoSuchElementException) [inline, virtual]`

Retrieves and removes the head of this queue.

This method differs from poll only in that it throws an exception if this queue is empty.

This implementation returns the result of poll unless the queue is empty.

Returns

a copy of the element in the head of the queue.

Exceptions

NoSuchElementException if the queue is empty.

Implements **decaf::util::Queue< E >** (p. 2510).

Reimplemented in **decaf::util::PriorityQueue< E >** (p. 2419).

References **decaf::util::Queue< E >::poll()**.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/AbstractQueue.h`

6.5 decaf::util::AbstractSequentialList< E > Class Template Reference

This class provides a skeletal implementation of the **List** (p. 1865) interface to minimize the effort required to implement this interface backed by a "sequential access" data store (such as a linked list).

```
#include <src/main/decaf/util/AbstractSequentialList.h>
```

Inheritance diagram for **decaf::util::AbstractSequentialList< E >**:

Public Member Functions

- `virtual ~AbstractSequentialList ()`

6.5.1 Detailed Description

```
template<typename E> class decaf::util::AbstractSequentialList< E >
```

This class provides a skeletal implementation of the **List** (p. 1865) interface to minimize the effort required to implement this interface backed by a "sequential access" data store (such as a linked list). For random access data (such as an array), **AbstractList** (p. 131) should be used in preference to this class.

This class is the opposite of the **AbstractList** (p. 131) class in the sense that it implements the "random access" methods (`get(int index)`, `set(int index, E element)`, `add(int index, E element)` and `remove(int index)`) on top of the list's list iterator, instead of the other way around.

To implement a list the programmer needs only to extend this class and provide implementations for the list iterator and size methods. For an unmodifiable list, the programmer need only implement the list iterator's `hasNext`, `next`, `hasPrevious`, `previous` and `index` methods.

For a modifiable list the programmer should additionally implement the list iterator's `set` method. For a variable-size list the programmer should additionally implement the list iterator's `remove` and `add` methods.

The programmer should generally provide a void (no argument) and collection constructor, as per the recommendation in the **Collection** (p. 982) interface specification.

Since

1.0

6.5.2 Constructor & Destructor Documentation

6.5.2.1 `template<typename E > virtual decaf::util::AbstractSequentialList< E >::~~AbstractSequentialList () [inline, virtual]`

The documentation for this class was generated from the following file:

- `src/main/decaf/util/AbstractSequentialList.h`

6.6 decaf::util::AbstractSet< E > Class Template Reference

This class provides a skeletal implementation of the **Set** (p. 2729) interface to minimize the effort required to implement this interface.

```
#include <src/main/decaf/util/AbstractSet.h>
```

Inheritance diagram for `decaf::util::AbstractSet< E >`:

Public Member Functions

- `virtual ~AbstractSet ()`
- `virtual bool removeAll (const Collection< E > &collection) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException)`

Removes from this set all of its elements that are contained in the specified collection (optional operation).

6.6.1 Detailed Description

`template<typename E> class decaf::util::AbstractSet< E >`

This class provides a skeletal implementation of the **Set** (p. 2729) interface to minimize the effort required to implement this interface. The process of implementing a set by extending this class is identical to that of implementing a **Collection** (p. 982) by extending **AbstractCollection** (p. 119), except that all of the methods and constructors in subclasses of this class must obey the additional constraints imposed by the **Set** (p. 2729) interface (for instance, the add method must not permit addition of multiple instances of an object to a set).

Since

1.0

6.6.2 Constructor & Destructor Documentation

6.6.2.1 `template<typename E> virtual decaf::util::AbstractSet< E >::~~AbstractSet () [inline, virtual]`

6.6.3 Member Function Documentation

6.6.3.1 `template<typename E> virtual bool decaf::util::AbstractSet< E >::removeAll (const Collection< E > & collection) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException) [inline, virtual]`

Removes from this set all of its elements that are contained in the specified collection (optional operation).

If the specified collection is also a set, this operation effectively modifies this set so that its value is the asymmetric set difference of the two sets.

This implementation determines which is the smaller of this set and the specified collection, by invoking the size method on each. If this set has fewer elements, then the implementation iterates over this set, checking each element returned by the iterator in turn to see if it is contained in the specified collection. If it is so contained, it is removed from this set with the iterator's remove method. If the specified collection has fewer elements, then the implementation iterates over the specified collection, removing from this set each element returned by the iterator, using this set's remove method.

Note that this implementation will throw an UnsupportedOperationException if the iterator returned by the iterator method does not implement the remove method.

Parameters

collection - The **Collection** (p. 982) whose elements are to be retained

Returns

true if the collection changed as a result of this call

Exceptions

UnsupportedOperationException

IllegalArgumentException

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 128).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/AbstractSet.h`

6.7 activemq::transport::AbstractTransportFactory Class Reference

Abstract implementation of the **TransportFactory** (p. 3071) interface, providing the base functionality that's common to most of the **TransportFactory** (p. 3071) instances.

```
#include <src/main/activemq/transport/AbstractTransportFactory.h>
```

Inheritance diagram for `activemq::transport::AbstractTransportFactory`:

Public Member Functions

- `virtual ~AbstractTransportFactory ()`

Protected Member Functions

- `virtual Pointer< wireformat::WireFormat > createWireFormat (const decaf::util::Properties &properties) throw (decaf::lang::exceptions::NoSuchElementException)`

*Creates the WireFormat that is configured for this **Transport** (p. 3066) and returns it.*

6.7.1 Detailed Description

Abstract implementation of the **TransportFactory** (p. 3071) interface, providing the base functionality that's common to most of the **TransportFactory** (p. 3071) instances.

Since

3.0

6.7.2 Constructor & Destructor Documentation

- 6.7.2.1** `virtual activemq::transport::AbstractTransportFactory::~~AbstractTransportFactory () [inline, virtual]`

6.7.3 Member Function Documentation

- 6.7.3.1** `virtual Pointer<wireformat::WireFormat> activemq::transport::AbstractTransportFactory::createWireFormat (const decaf::util::Properties & properties) throw (decaf::lang::exceptions::NoSuchElementException) [protected, virtual]`

Creates the WireFormat that is configured for this **Transport** (p. 3066) and returns it.

The default WireFormat is Openwire.

Parameters

properties The properties that were configured on the URI.

Returns

a pointer to a WireFormat instance that the caller then owns.

Exceptions

NoSuchElementException if the configured WireFormat is not found.

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/AbstractTransportFactory.h`

6.8 activemq::core::ActiveMQAckHandler Class Reference

Interface class that is used to give CMS Messages an interface to Ack themselves with.

```
#include <src/main/activemq/core/ActiveMQAckHandler.h>
```

Public Member Functions

- virtual `~ActiveMQAckHandler()`
- virtual void `acknowledgeMessage` (const `commands::Message *message`)=0 throw (`cms::CMSEException`)

Method called to acknowledge the message passed.

6.8.1 Detailed Description

Interface class that is used to give CMS Messages an interface to Ack themselves with.

Since

2.0

6.8.2 Constructor & Destructor Documentation

- 6.8.2.1 virtual `activemq::core::ActiveMQAckHandler::~ActiveMQAckHandler ()`
[inline, virtual]

6.8.3 Member Function Documentation

- 6.8.3.1 virtual void `activemq::core::ActiveMQAckHandler::acknowledgeMessage` (`const commands::Message * message`) throw (`cms::CMSEException`)
[pure virtual]

Method called to acknowledge the message passed.

Parameters

message Message to Acknowledge

Exceptions

CMSEException

The documentation for this class was generated from the following file:

- `src/main/activemq/core/ActiveMQAckHandler.h`

6.9 activemq::commands::ActiveMQBlobMessage Class Reference

```
#include <src/main/activemq/commands/ActiveMQBlobMessage.h>
```

Inheritance diagram for `activemq::commands::ActiveMQBlobMessage`:

Public Member Functions

- **ActiveMQBlobMessage** ()
- virtual **~ActiveMQBlobMessage** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **ActiveMQBlobMessage * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1372) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent.*
- virtual **cms::Message * clone** () const
Clone this message exactly, returns a new instance that the caller is required to delete.
- std::string **getRemoteBlobUrl** () const
Get the Remote URL of the Blob.
- void **setRemoteBlobUrl** (const std::string &remoteURL)
Set the Remote URL of the Blob.
- std::string **getMimeType** () const
Get the Mime Type of the Blob.
- void **setMimeType** (const std::string &mimeType)
Set the Mime Type of the Blob.
- std::string **getName** () const
Gets the Name of the Blob.
- void **setName** (const std::string &name)
Sets the Name of the Blob.

- bool **isDeletedByBroker** () const
Gets if this Blob is deleted by the Broker.
- void **setDeletedByBroker** (bool value)
Sets the Deleted By Broker flag.

Static Public Attributes

- static const unsigned char **ID _ACTIVEMQBLOBMESSAGE** = 29
- static const std::string **BINARY _MIME _TYPE**

6.9.1 Constructor & Destructor Documentation

6.9.1.1 `activemq::commands::ActiveMQBlobMessage::ActiveMQBlobMessage ()`

6.9.1.2 `virtual
activemq::commands::ActiveMQBlobMessage::~~ActiveMQBlobMessage ()` [inline, virtual]

6.9.2 Member Function Documentation

6.9.2.1 `virtual cms::Message* activemq::commands::ActiveMQBlobMessage::clone () const` [inline, virtual]

Clone this message exactly, returns a new instance that the caller is required to delete.

Returns

new copy of this message

6.9.2.2 `virtual ActiveMQBlobMessage* activemq::commands::ActiveMQBlobMessage::cloneDataStructure () const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Reimplemented from `activemq::commands::Message` (p. 2023).

6.9.2.3 `virtual void activemq::commands::ActiveMQBlobMessage::copyDataStructure (const DataStructure * src)` [virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Reimplemented from `activemq::commands::Message` (p. 2023).

6.9.2.4 `virtual bool activemq::commands::ActiveMQBlobMessage::equals (const DataStructure * value) const` [virtual]

Compares the `DataStructure` (p. 1372) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if `DataStructure`'s are Equal.

Reimplemented from `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 331).

6.9.2.5 `virtual unsigned char activemq::commands::ActiveMQBlobMessage::getDataStructureType () const` [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns

new `DataStructure` (p. 1372) type copy.

Reimplemented from `activemq::commands::Message` (p. 2025).

6.9.2.6 `std::string activemq::commands::ActiveMQBlobMessage::getMimeType () const` [inline]

Get the Mime Type of the Blob.

Returns

string holding the MIME Type.

6.9.2.7 `std::string activemq::commands::ActiveMQBlobMessage::getName () const` [inline]

Gets the Name of the Blob.

Returns

string name of the Blob.

6.9.2.8 `std::string activemq::commands::ActiveMQBlobMessage::getRemoteBlobUrl () const` [inline]

Get the Remote URL of the Blob.

Returns

string from of the Remote Blob URL.

6.9.2.9 `bool activemq::commands::ActiveMQBlobMessage::isDeletedByBroker () const` [inline]

Gets if this Blob is deleted by the Broker.

Returns

true if the Blob is deleted by the Broker.

6.9.2.10 `void activemq::commands::ActiveMQBlobMessage::setDeletedByBroker (bool value)` [inline]

Sets the Deleted By Broker flag.

Parameters

value - set the Delete by broker flag to value.

6.9.2.11 `void activemq::commands::ActiveMQBlobMessage::setMimeType (const std::string & mimeType)` [inline]

Set the Mime Type of the Blob.

Parameters

mimeType - String holding the MIME Type.

6.9.2.12 `void activemq::commands::ActiveMQBlobMessage::setName (const std::string & name)` [inline]

Sets the Name of the Blob.

Parameters

name - Name of the Blob.

6.9.2.13 `void activemq::commands::ActiveMQBlobMessage::setRemoteBlobUrl (const std::string & remoteURL) [inline]`

Set the Remote URL of the Blob.

Parameters

remoteURL - String form of the Remote URL.

6.9.2.14 `virtual std::string activemq::commands::ActiveMQBlobMessage::toString () const [virtual]`

Returns a string containing the information for this **DataStructure** (p.1372) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from `activemq::commands::Message` (p. 2033).

6.9.3 Field Documentation

6.9.3.1 `const std::string activemq::commands::ActiveMQBlobMessage::BINARY_MIME_TYPE [static]`

6.9.3.2 `const unsigned char activemq::commands::ActiveMQBlobMessage::ID_ACTIVEMQBLOBMESSAGE = 29 [static]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ActiveMQBlobMessage.h`

6.10 `activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller` Class Reference

Marshaling code for Open Wire Format for `ActiveMQBlobMessageMarshaller` (p. 146).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQBlobMessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller`:

Public Member Functions

- `ActiveMQBlobMessageMarshaller ()`
- `virtual ~ActiveMQBlobMessageMarshaller ()`
- `virtual commands::DataStructure * createObject () const`

Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)

Un-marshall an object instance from the data input stream.

- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshall an object instance from the data input stream.

- virtual void **looseMarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.10.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p.146).
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.10.2 Constructor & Destructor Documentation

6.10.2.1 `activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller::ActiveMQBlobMessageMarshaller () [inline]`

6.10.2.2 `virtual activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller::~~ActiveMQBlobMessageMarshaller () [inline, virtual]`

6.10.3 Member Function Documentation

6.10.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.10.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.10.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2176).

6.10.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2176).

6.10.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2177).

6.10.3.6 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2178).

```
6.10.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller::tightUnmars
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2178).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQBlobMessageMarshaller.h`

6.11 `activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller` Class Reference

Marshaling code for Open Wire Format for `ActiveMQBlobMessageMarshaller` (p. 150).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQBlobMessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller`:

Public Member Functions

- **ActiveMQBlobMessageMarshaller** ()
- virtual **~ActiveMQBlobMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.11.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p.150).
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.11.2 Constructor & Destructor Documentation

6.11.2.1 `activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller::ActiveMQBlobMessageMarshaller () [inline]`

6.11.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller::~~ActiveMQBlobMessageMarshaller () [inline, virtual]`

6.11.3 Member Function Documentation

6.11.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.11.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.11.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2184).

6.11.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2184).

6.11.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2185).

6.11.3.6 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2186).

```
6.11.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller::tightUnmars
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2186).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQBlobMessageMarshaller.h`

6.12 `activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller` Class Reference

Marshaling code for Open Wire Format for `ActiveMQBlobMessageMarshaller` (p. 154).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQBlobMessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller`:

Public Member Functions

- **ActiveMQBlobMessageMarshaller** ()
- virtual **~ActiveMQBlobMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.12.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p.154).
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.12.2 Constructor & Destructor Documentation

6.12.2.1 `activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller::ActiveMQBlobMessageMarshaller () [inline]`

6.12.2.2 `virtual activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller::~~ActiveMQBlobMessageMarshaller () [inline, virtual]`

6.12.3 Member Function Documentation

6.12.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.12.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.12.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2172).

6.12.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2172).

6.12.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2173).

6.12.3.6 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2174).

```
6.12.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller::tightUnmars
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2174).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQBlobMessageMarshaller.h`

6.13 `activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller` Class Reference

Marshaling code for Open Wire Format for `ActiveMQBlobMessageMarshaller` (p. 158).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQBlobMessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller`:

Public Member Functions

- **ActiveMQBlobMessageMarshaller** ()
- virtual **~ActiveMQBlobMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.13.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p.158).
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.13.2 Constructor & Destructor Documentation

6.13.2.1 `activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller::ActiveMQBlobMessageMarshaller () [inline]`

6.13.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller::~~ActiveMQBlobMessageMarshaller () [inline, virtual]`

6.13.3 Member Function Documentation

6.13.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.13.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.13.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2180).

6.13.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2180).

6.13.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2181).

6.13.3.6 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2182).

```
6.13.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller::tightUnmars
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2182).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQBlobMessageMarshaller.h`

6.14 `activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller` Class Reference

Marshaling code for Open Wire Format for `ActiveMQBlobMessageMarshaller` (p. 162).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQBlobMessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller`:

Public Member Functions

- **ActiveMQBlobMessageMarshaller** ()
- virtual **~ActiveMQBlobMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.14.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p.162).
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.14.2 Constructor & Destructor Documentation

6.14.2.1 `activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller::ActiveMQBlobMessageMarshaller () [inline]`

6.14.2.2 `virtual activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller::~~ActiveMQBlobMessageMarshaller () [inline, virtual]`

6.14.3 Member Function Documentation

6.14.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.14.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.14.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2167).

6.14.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2168).

6.14.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2169).

6.14.3.6 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions

- IOException* if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2169).

```
6.14.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller::tightUnmars
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2170).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQBlobMessageMarshaller.h`

6.15 `activemq::commands::ActiveMQBytesMessage` Class Reference

```
#include <src/main/activemq/commands/ActiveMQBytesMessage.h>
```

Inheritance diagram for `activemq::commands::ActiveMQBytesMessage`:

Public Member Functions

- **ActiveMQBytesMessage** ()
- virtual **~ActiveMQBytesMessage** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **ActiveMQBytesMessage * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1372) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent.*
- virtual **cms::BytesMessage * clone** () const
Clone this message exactly, returns a new instance that the caller is required to delete.
- virtual void **clearBody** () throw (cms::CMSEException)
Clears out the body of the message.
- virtual void **onSend** ()
Store the Data that was written to the stream before a send.
- virtual void **setBodyBytes** (const unsigned char *buffer, std::size_t numBytes) throw (cms::MessageNotWriteableException, cms::CMSEException)
sets the bytes given to the message body.
- virtual unsigned char * **getBodyBytes** () const throw (cms::MessageNotReadableException, cms::CMSEException)
Gets the bytes that are contained in this message, user should copy this data into a user allocated buffer.
- virtual std::size_t **getBodyLength** () const throw (cms::MessageNotReadableException, cms::CMSEException)
Returns the number of bytes contained in the body of this message.
- virtual void **reset** () throw (cms::MessageFormatException, cms::CMSEException)
Puts the message body in read-only mode and repositions the stream of bytes to the beginning.
- virtual bool **readBoolean** () const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException)
Reads a Boolean from the Bytes message stream.

- virtual void **writeBoolean** (bool value) throw (cms::MessageNotWriteableException, cms::CMSEException)
Writes a boolean to the bytes message stream as a 1-byte value.
- virtual unsigned char **readByte** () const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException)
Reads a Byte from the Bytes message stream.
- virtual void **writeByte** (unsigned char value) throw (cms::MessageNotWriteableException, cms::CMSEException)
Writes a byte to the bytes message stream as a 1-byte value.
- virtual std::size_t **readBytes** (std::vector< unsigned char > &value) const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException)
Reads a byte array from the bytes message stream.
- virtual void **writeBytes** (const std::vector< unsigned char > &value) throw (cms::MessageNotWriteableException, cms::CMSEException)
Writes a byte array to the bytes message stream using the vector size as the number of bytes to write.
- virtual std::size_t **readBytes** (unsigned char *buffer, std::size_t length) const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException)
Reads a portion of the bytes message stream.
- virtual void **writeBytes** (const unsigned char *value, std::size_t offset, std::size_t length) throw (cms::MessageNotWriteableException, cms::CMSEException)
Writes a portion of a byte array to the bytes message stream.
- virtual char **readChar** () const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException)
Reads a Char from the Bytes message stream.
- virtual void **writeChar** (char value) throw (cms::MessageNotWriteableException, cms::CMSEException)
Writes a char to the bytes message stream as a 1-byte value.
- virtual float **readFloat** () const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException)
Reads a 32 bit float from the Bytes message stream.
- virtual void **writeFloat** (float value) throw (cms::MessageNotWriteableException, cms::CMSEException)
Writes a float to the bytes message stream as a 4 byte value.
- virtual double **readDouble** () const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException)
Reads a 64 bit double from the Bytes message stream.
- virtual void **writeDouble** (double value) throw (cms::MessageNotWriteableException, cms::CMSEException)

Writes a double to the bytes message stream as a 8 byte value.

- virtual short **readShort** () const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException)

Reads a 16 bit signed short from the Bytes message stream.

- virtual void **writeShort** (short value) throw (cms::MessageNotWriteableException, cms::CMSEException)

Writes a signed short to the bytes message stream as a 2 byte value.

- virtual unsigned short **readUnsignedShort** () const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException)

Reads a 16 bit unsigned short from the Bytes message stream.

- virtual void **writeUnsignedShort** (unsigned short value) throw (cms::MessageNotWriteableException, cms::CMSEException)

Writes a unsigned short to the bytes message stream as a 2 byte value.

- virtual int **readInt** () const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException)

Reads a 32 bit signed integer from the Bytes message stream.

- virtual void **writeInt** (int value) throw (cms::MessageNotWriteableException, cms::CMSEException)

Writes a signed int to the bytes message stream as a 4 byte value.

- virtual long long **readLong** () const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException)

Reads a 64 bit long from the Bytes message stream.

- virtual void **writeLong** (long long value) throw (cms::MessageNotWriteableException, cms::CMSEException)

Writes a long long to the bytes message stream as a 8 byte value.

- virtual std::string **readString** () const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException)

Reads an ASCII String from the Bytes message stream.

- virtual void **writeString** (const std::string &value) throw (cms::MessageNotWriteableException, cms::CMSEException)

Writes an ASCII String to the Bytes message stream.

- virtual std::string **readUTF** () const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException)

Reads an UTF String from the BytesMessage stream.

- virtual void **writeUTF** (const std::string &value) throw (cms::MessageNotWriteableException, cms::CMSEException)

Writes an UTF String to the BytesMessage stream.

Static Public Attributes

- static const unsigned char `ID_ACTIVEMQBYTESMESSAGE` = 24

6.15.1 Constructor & Destructor Documentation

6.15.1.1 `activemq::commands::ActiveMQBytesMessage::ActiveMQBytesMessage ()`

6.15.1.2 `virtual activemq::commands::ActiveMQBytesMessage::~~ActiveMQBytesMessage () [virtual]`

6.15.2 Member Function Documentation

6.15.2.1 `virtual void activemq::commands::ActiveMQBytesMessage::clearBody () throw (cms::CMSException) [virtual]`

Clears out the body of the message.

This does not clear the headers or properties.

Reimplemented from `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 331).

6.15.2.2 `virtual cms::BytesMessage* activemq::commands::ActiveMQBytesMessage::clone () const [inline, virtual]`

Clone this message exactly, returns a new instance that the caller is required to delete.

Returns

new copy of this message

6.15.2.3 `virtual ActiveMQBytesMessage* activemq::commands::ActiveMQBytesMessage::cloneDataStructure () const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Reimplemented from `activemq::commands::Message` (p. 2023).

6.15.2.4 `virtual void activemq::commands::ActiveMQBytesMessage::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Reimplemented from `activemq::commands::Message` (p. 2023).

6.15.2.5 `virtual bool activemq::commands::ActiveMQBytesMessage::equals (const DataStructure * value) const [virtual]`

Compares the `DataStructure` (p. 1372) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Parameters

value The `Command` (p. 991) to compare to this one.

Returns

true if `DataStructure`'s are Equal.

Reimplemented from `activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 331).

6.15.2.6 `virtual unsigned char* activemq::commands::ActiveMQBytesMessage::getBodyBytes () const throw (cms::MessageNotReadableException, cms::CMSException) [virtual]`

Gets the bytes that are contained in this message, user should copy this data into a user allocated buffer.

Call `getBodyLength` to determine the number of bytes to expect.

Returns

const pointer to a byte buffer

Exceptions

CMSException - If an internal error occurs.

MessageNotReadableException - If the message is in Write Only Mode.

6.15.2.7 `virtual std::size_t activemq::commands::ActiveMQBytesMessage::getBodyLength () const throw (cms::MessageNotReadableException, cms::CMSException) [virtual]`

Returns the number of bytes contained in the body of this message.

Returns

number of bytes.

Exceptions

CMSException - If an internal error occurs.

MessageNotReadableException - If the message is in Write Only Mode.

6.15.2.8 `virtual unsigned char activemq::commands::ActiveMQBytesMessage::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataStructure** (p. 1372) type copy.

Reimplemented from **activemq::commands::Message** (p. 2025).

6.15.2.9 `virtual void activemq::commands::ActiveMQBytesMessage::onSend () [virtual]`

Store the Data that was written to the stream before a send.

Reimplemented from **activemq::commands::ActiveMQMessageTemplate<cms::BytesMessage>** (p. 338).

6.15.2.10 `virtual bool activemq::commands::ActiveMQBytesMessage::readBoolean () const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSException) [virtual]`

Reads a Boolean from the Bytes message stream.

Returns

boolean value from stream

Exceptions

CMSException - if the CMS provider fails to read the message due to some internal error.

MessageEOFException - if unexpected end of bytes stream has been reached.

MessageNotReadableException - if the message is in write-only mode.

6.15.2.11 `virtual unsigned char activemq::commands::ActiveMQBytesMessage::readByte () const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSException) [virtual]`

Reads a Byte from the Bytes message stream.

Returns

unsigned char value from stream

Exceptions

CMSException - if the CMS provider fails to read the message due to some internal error.

MessageEOFException - if unexpected end of bytes stream has been reached.

MessageNotReadableException - if the message is in write-only mode.

6.15.2.12 `virtual std::size_t activemq::commands::ActiveMQBytesMessage::readBytes (std::vector< unsigned char > & value) const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSException) [virtual]`

Reads a byte array from the bytes message stream.

If the length of vector value is less than the number of bytes remaining to be read from the stream, the vector should be filled. A subsequent call reads the next increment, and so on.

If the number of bytes remaining in the stream is less than the length of vector value, the bytes should be read into the vector. The return value of the total number of bytes read will be less than the length of the vector, indicating that there are no more bytes left to be read from the stream. The next read of the stream returns -1.

Parameters

value buffer to place data in

Returns

the total number of bytes read into the buffer, or -1 if there is no more data because the end of the stream has been reached

Exceptions

CMSException - if the CMS provider fails to read the message due to some internal error.

MessageEOFException - if unexpected end of bytes stream has been reached.

MessageNotReadableException - if the message is in write-only mode.

6.15.2.13 `virtual std::size_t activemq::commands::ActiveMQBytesMessage::readBytes (unsigned char * buffer, std::size_t length) const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSException) [virtual]`

Reads a portion of the bytes message stream.

If the length of array value is less than the number of bytes remaining to be read from the stream, the array should be filled. A subsequent call reads the next increment, and so on.

If the number of bytes remaining in the stream is less than the length of array value, the bytes should be read into the array. The return value of the total number of bytes read will be less than the length of the array, indicating that there are no more bytes left to be read from the stream. The next read of the stream returns -1.

If length is negative, or length is greater than the length of the array value, then an `IndexOutOfBoundsException` is thrown. No bytes will be read from the stream for this exception case.

Parameters

buffer the buffer into which the data is read

length the number of bytes to read; must be less than or equal to value.length

Returns

the total number of bytes read into the buffer, or -1 if there is no more data because the end of the stream has been reached

Exceptions

CMSEException - if the CMS provider fails to read the message due to some internal error.

MessageEOFException - if unexpected end of bytes stream has been reached.

MessageNotReadableException - if the message is in write-only mode.

6.15.2.14 virtual char activemq::commands::ActiveMQBytesMessage::readChar
() const throw (cms::MessageEOFException,
cms::MessageNotReadableException, cms::CMSEException) [virtual]

Reads a Char from the Bytes message stream.

Returns

char value from stream

Exceptions

CMSEException - if the CMS provider fails to read the message due to some internal error.

MessageEOFException - if unexpected end of bytes stream has been reached.

MessageNotReadableException - if the message is in write-only mode.

6.15.2.15 virtual double ac-
tivismq::commands::ActiveMQBytesMessage::readDouble
() const throw (cms::MessageEOFException,
cms::MessageNotReadableException, cms::CMSEException) [virtual]

Reads a 64 bit double from the Bytes message stream.

Returns

double value from stream

Exceptions

CMSEException - if the CMS provider fails to read the message due to some internal error.

MessageEOFException - if unexpected end of bytes stream has been reached.

MessageNotReadableException - if the message is in write-only mode.

6.15.2.16 virtual float activemq::commands::ActiveMQBytesMessage::readFloat
() const throw (cms::MessageEOFException,
cms::MessageNotReadableException, cms::CMSException) [virtual]

Reads a 32 bit float from the Bytes message stream.

Returns

double value from stream

Exceptions

CMSException - if the CMS provider fails to read the message due to some internal error.

MessageEOFException - if unexpected end of bytes stream has been reached.

MessageNotReadableException - if the message is in write-only mode.

6.15.2.17 virtual int activemq::commands::ActiveMQBytesMessage::readInt
() const throw (cms::MessageEOFException,
cms::MessageNotReadableException, cms::CMSException) [virtual]

Reads a 32 bit signed integer from the Bytes message stream.

Returns

int value from stream

Exceptions

CMSException - if the CMS provider fails to read the message due to some internal error.

MessageEOFException - if unexpected end of bytes stream has been reached.

MessageNotReadableException - if the message is in write-only mode.

6.15.2.18 virtual long long ac-
tivemq::commands::ActiveMQBytesMessage::readLong
() const throw (cms::MessageEOFException,
cms::MessageNotReadableException, cms::CMSException) [virtual]

Reads a 64 bit long from the Bytes message stream.

Returns

long long value from stream

Exceptions

CMSException - if the CMS provider fails to read the message due to some internal error.

MessageEOFException - if unexpected end of bytes stream has been reached.

MessageNotReadableException - if the message is in write-only mode.

6.15.2.19 virtual short activemq::commands::ActiveMQBytesMessage::readShort
() const throw (cms::MessageEOFException,
cms::MessageNotReadableException, cms::CMSException) [virtual]

Reads a 16 bit signed short from the Bytes message stream.

Returns

short value from stream

Exceptions

CMSException - if the CMS provider fails to read the message due to some internal error.

MessageEOFException - if unexpected end of bytes stream has been reached.

MessageNotReadableException - if the message is in write-only mode.

6.15.2.20 virtual std::string ac-
tivismq::commands::ActiveMQBytesMessage::readString
() const throw (cms::MessageEOFException,
cms::MessageNotReadableException, cms::CMSException) [virtual]

Reads an ASCII String from the Bytes message stream.

Returns

String from stream

Exceptions

CMSException - if the CMS provider fails to read the message due to some internal error.

MessageEOFException - if unexpected end of bytes stream has been reached.

MessageNotReadableException - if the message is in write-only mode.

6.15.2.21 virtual unsigned short ac-
tivismq::commands::ActiveMQBytesMessage::readUnsignedShort
() const throw (cms::MessageEOFException,
cms::MessageNotReadableException, cms::CMSException) [virtual]

Reads a 16 bit unsigned short from the Bytes message stream.

Returns

unsigned short value from stream

Exceptions

CMSException - if the CMS provider fails to read the message due to some internal error.

MessageEOFException - if unexpected end of bytes stream has been reached.

MessageNotReadableException - if the message is in write-only mode.

6.15.2.22 virtual std::string activemq::commands::ActiveMQBytesMessage::readUTF () const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSException) [virtual]

Reads an UTF String from the BytesMessage stream.

Returns

String from stream

Exceptions

CMSException - if the CMS provider fails to read the message due to some internal error.

MessageEOFException - if unexpected end of bytes stream has been reached.

MessageNotReadableException - if the message is in write-only mode.

6.15.2.23 virtual void activemq::commands::ActiveMQBytesMessage::reset () throw (cms::MessageFormatException, cms::CMSException) [virtual]

Puts the message body in read-only mode and repositions the stream of bytes to the beginning.

Exceptions

CMSException - If the provider fails to perform the reset operation.

MessageFormatException - If the **Message** (p. 2018) has an invalid format.

6.15.2.24 virtual void activemq::commands::ActiveMQBytesMessage::setBodyBytes (const unsigned char * *buffer*, std::size_t *numBytes*) throw (cms::MessageNotWritableException, cms::CMSException) [virtual]

sets the bytes given to the message body.

Parameters

buffer Byte Buffer to copy

numBytes Number of bytes in Buffer to copy

Exceptions

CMSException - If an internal error occurs.

MessageNotWritableException - if in Read Only Mode.

6.15.2.25 virtual std::string activemq::commands::ActiveMQBytesMessage::toString () const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1372) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from `activemq::commands::Message` (p. 2033).

6.15.2.26 `virtual void activemq::commands::ActiveMQBytesMessage::writeBoolean (bool value) throw (cms::MessageNotWriteableException, cms::CMSException) [virtual]`

Writes a boolean to the bytes message stream as a 1-byte value.

The value true is written as the value (byte)1; the value false is written as the value (byte)0.

Parameters

value boolean to write to the stream

Exceptions

CMSException - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException - if the message is in read-only mode.

6.15.2.27 `virtual void activemq::commands::ActiveMQBytesMessage::writeByte (unsigned char value) throw (cms::MessageNotWriteableException, cms::CMSException) [virtual]`

Writes a byte to the bytes message stream as a 1-byte value.

Parameters

value byte to write to the stream

Exceptions

CMSException - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException - if the message is in read-only mode.

6.15.2.28 `virtual void activemq::commands::ActiveMQBytesMessage::writeBytes (const unsigned char * value, std::size_t offset, std::size_t length) throw (cms::MessageNotWriteableException, cms::CMSException) [virtual]`

Writes a portion of a byte array to the bytes message stream.

size as the number of bytes to write.

Parameters

value bytes to write to the stream

offset the initial offset within the byte array

length the number of bytes to use

Exceptions

CMSException - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException - if the message is in read-only mode.

6.15.2.29 virtual void activemq::commands::ActiveMQBytesMessage::writeBytes
(const std::vector< unsigned char > & *value*) throw (cms::MessageNotWriteableException, cms::CMSException) [virtual]

Writes a byte array to the bytes message stream using the vector size as the number of bytes to write.

Parameters

value bytes to write to the stream

Exceptions

CMSException - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException - if the message is in read-only mode.

6.15.2.30 virtual void activemq::commands::ActiveMQBytesMessage::writeChar
(char *value*) throw (cms::MessageNotWriteableException, cms::CMSException) [virtual]

Writes a char to the bytes message stream as a 1-byte value.

Parameters

value char to write to the stream

Exceptions

CMSException - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException - if the message is in read-only mode.

6.15.2.31 virtual void activemq::commands::ActiveMQBytesMessage::writeDouble
(double *value*) throw (cms::MessageNotWriteableException, cms::CMSException) [virtual]

Writes a double to the bytes message stream as a 8 byte value.

Parameters

value double to write to the stream

Exceptions

CMSException - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException - if the message is in read-only mode.

6.15.2.32 virtual void activemq::commands::ActiveMQBytesMessage::writeFloat
(float *value*) throw (cms::MessageNotWriteableException,
cms::CMSException) [virtual]

Writes a float to the bytes message stream as a 4 byte value.

Parameters

value float to write to the stream

Exceptions

CMSException - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException - if the message is in read-only mode.

6.15.2.33 virtual void activemq::commands::ActiveMQBytesMessage::writeInt
(int *value*) throw (cms::MessageNotWriteableException,
cms::CMSException) [virtual]

Writes a signed int to the bytes message stream as a 4 byte value.

Parameters

value signed int to write to the stream

Exceptions

CMSException - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException - if the message is in read-only mode.

6.15.2.34 virtual void activemq::commands::ActiveMQBytesMessage::writeLong
(long long *value*) throw (cms::MessageNotWriteableException,
cms::CMSException) [virtual]

Writes a long long to the bytes message stream as a 8 byte value.

Parameters

value signed long long to write to the stream

Exceptions

CMSException - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException - if the message is in read-only mode.

6.15.2.35 virtual void activemq::commands::ActiveMQBytesMessage::writeShort
(short *value*) throw (cms::MessageNotWriteableException,
cms::CMSException) [virtual]

Writes a signed short to the bytes message stream as a 2 byte value.

Parameters

value signed short to write to the stream

Exceptions

CMSEException - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException - if the message is in read-only mode.

6.15.2.36 virtual void activemq::commands::ActiveMQBytesMessage::writeString (const std::string & *value*) throw (cms::MessageNotWriteableException, cms::CMSEException) [virtual]

Writes an ASCII String to the Bytes message stream.

Parameters

value String to write to the stream

Exceptions

CMSEException - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException - if the message is in read-only mode.

6.15.2.37 virtual void activemq::commands::ActiveMQBytesMessage::writeUnsignedShort (unsigned short *value*) throw (cms::MessageNotWriteableException, cms::CMSEException) [virtual]

Writes a unsigned short to the bytes message stream as a 2 byte value.

Parameters

value unsigned short to write to the stream

Exceptions

CMSEException - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException - if the message is in read-only mode.

6.15.2.38 virtual void activemq::commands::ActiveMQBytesMessage::writeUTF (const std::string & *value*) throw (cms::MessageNotWriteableException, cms::CMSEException) [virtual]

Writes an UTF String to the BytesMessage stream.

Parameters

value String to write to the stream

Exceptions

CMSEException - if the CMS provider fails to read the message due to some internal error.

MessageEOFException - if unexpected end of bytes stream has been reached.

MessageNotReadableException - if the message is in write-only mode.

6.15.3 Field Documentation

6.15.3.1 `const unsigned char activemq::commands::ActiveMQBytesMessage::ID - ACTIVEMQBYTESMESSAGE = 24` [static]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ActiveMQBytesMessage.h`

6.16 `activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller` Class Reference

Marshaling code for Open Wire Format for `ActiveMQBytesMessageMarshaller` (p. 182).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQBytesMessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller`:

Public Member Functions

- `ActiveMQBytesMessageMarshaller ()`
- `virtual ~ActiveMQBytesMessageMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaller.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshall an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write a object instance to data output stream.
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)`
Un-marshall an object instance from the data input stream.

6.16

activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller

Class Reference

185

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.16.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 182).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.16.2 Constructor & Destructor Documentation

6.16.2.1 **activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller::ActiveMQBytesMessageMarshaller** () [inline]

6.16.2.2 **virtual**
activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller::~~ActiveMQBytesMessageMarshaller () [inline, virtual]

6.16.3 Member Function Documentation

6.16.3.1 **virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller::createObject** () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1331).

6.16.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller::getDataStructureType** () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1336).

6.16.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2176).

6.16.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2176).

6.16.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

6.16

activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller

Class Reference

187

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2177).

6.16.3.6 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2178).

6.16.3.7 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2178).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQBytesMessageMarshaller.h`

6.17 `activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller` Class Reference

Marshaling code for Open Wire Format for `ActiveMQBytesMessageMarshaller` (p. 186).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQBytesMessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller`:

Public Member Functions

- `ActiveMQBytesMessageMarshaller ()`
- `virtual ~ActiveMQBytesMessageMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaller.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write a object instance to data output stream.
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)`

6.17

activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller **Class Reference**

189

Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.17.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p.186).
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.17.2 Constructor & Destructor Documentation

6.17.2.1 **activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller::ActiveMQBytesMessageMarshaller ()** [inline]

6.17.2.2 **virtual activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller::~~ActiveMQBytesMessageMarshaller ()** [inline, virtual]

6.17.3 Member Function Documentation

6.17.3.1 **virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller::createObject () const** [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1331).

6.17.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller::getDataStructureType () const** [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1336).

6.17.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2184).

6.17.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2184).

6.17.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

6.17

activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller

Class Reference

191

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2185).

6.17.3.6 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller::tightMarshal
(OpenWireFormat * *wireFormat*, commands::DataStructure
* *dataStructure*, decaf::io::DataOutputStream * *dataOut*,
utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2186).

6.17.3.7 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller::tightUnmarshal
(OpenWireFormat * *wireFormat*, commands::DataStructure
* *dataStructure*, decaf::io::DataInputStream * *dataIn*,
utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2186).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQBytesMessageMarshaller.h`

6.18 `activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller` Class Reference

Marshaling code for Open Wire Format for **`ActiveMQBytesMessageMarshaller`** (p. 190).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQBytesMessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller`:

Public Member Functions

- **`ActiveMQBytesMessageMarshaller`** ()
- virtual **`~ActiveMQBytesMessageMarshaller`** ()
- virtual **`commands::DataStructure * createObject`** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **`getDataStructureType`** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **`tightUnmarshal`** (**`OpenWireFormat`** *wireFormat, **`commands::DataStructure`** *dataStructure, **`decaf::io::DataInputStream`** *dataIn, **`utils::BooleanStream`** *bs) throw (**`decaf::io::IOException`**)
Un-marshal an object instance from the data input stream.
- virtual int **`tightMarshal1`** (**`OpenWireFormat`** *wireFormat, **`commands::DataStructure`** *dataStructure, **`utils::BooleanStream`** *bs) throw (**`decaf::io::IOException`**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **`tightMarshal2`** (**`OpenWireFormat`** *wireFormat, **`commands::DataStructure`** *dataStructure, **`decaf::io::DataOutputStream`** *dataOut, **`utils::BooleanStream`** *bs) throw (**`decaf::io::IOException`**)
Write a object instance to data output stream.
- virtual void **`looseUnmarshal`** (**`OpenWireFormat`** *wireFormat, **`commands::DataStructure`** *dataStructure, **`decaf::io::DataInputStream`** *dataIn) throw (**`decaf::io::IOException`**)

6.18

activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller **Class Reference**

193

Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.18.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 190).
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.18.2 Constructor & Destructor Documentation

6.18.2.1 **activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller::ActiveMQBytesMessageMarshaller ()** [inline]

6.18.2.2 **virtual activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller::~~ActiveMQBytesMessageMarshaller ()** [inline, virtual]

6.18.3 Member Function Documentation

6.18.3.1 **virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller::createObject () const** [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1331).

6.18.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller::getDataStructureType () const** [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1336).

6.18.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2172).

6.18.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2172).

6.18.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

6.18

activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller

Class Reference

195

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2173).

6.18.3.6 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2174).

6.18.3.7 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2174).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQBytesMessageMarshaller.h`

6.19 `activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller` Class Reference

Marshaling code for Open Wire Format for `ActiveMQBytesMessageMarshaller` (p. 194).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQBytesMessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller`:

Public Member Functions

- `ActiveMQBytesMessageMarshaller ()`
- `virtual ~ActiveMQBytesMessageMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaller.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write a object instance to data output stream.
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)`

6.19

activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller Class Reference

197

Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.19.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 194).
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.19.2 Constructor & Destructor Documentation

6.19.2.1 activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller::ActiveMQBytesMessageMarshaller () [inline]

6.19.2.2 virtual
activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller::~~ActiveMQBytesMessageMarshaller () [inline, virtual]

6.19.3 Member Function Documentation

6.19.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1331).

6.19.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1336).

6.19.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2180).

6.19.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2180).

6.19.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

6.19

activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller

Class Reference

199

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2181).

6.19.3.6 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2182).

6.19.3.7 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller::tightUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2182).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQBytesMessageMarshaller.h`

6.20 `activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller` Class Reference

Marshaling code for Open Wire Format for `ActiveMQBytesMessageMarshaller` (p. 198).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQBytesMessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller`:

Public Member Functions

- `ActiveMQBytesMessageMarshaller ()`
- `virtual ~ActiveMQBytesMessageMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaller.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write a object instance to data output stream.
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)`

6.20

activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller Class Reference

201

Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.20.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p.198).
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.20.2 Constructor & Destructor Documentation

6.20.2.1 activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller::ActiveMQBytesMessageMarshaller () [inline]

6.20.2.2 virtual activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller::~~ActiveMQBytesMessageMarshaller () [inline, virtual]

6.20.3 Member Function Documentation

6.20.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1331).

6.20.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1336).

6.20.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2167).

6.20.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2168).

6.20.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

6.20

activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller

Class Reference

203

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2169).

6.20.3.6 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2169).

6.20.3.7 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2170).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQBytesMessageMarshaller.h`

6.21 activemq::core::ActiveMQConnection Class Reference

Concrete connection used for all connectors to the ActiveMQ broker.

```
#include <src/main/activemq/core/ActiveMQConnection.h>
```

Inheritance diagram for `activemq::core::ActiveMQConnection`:

Public Member Functions

- **ActiveMQConnection** (const **Pointer**< **transport::Transport** > &transport, const **Pointer**< **decaf::util::Properties** > &properties)
Constructor.
- virtual **~ActiveMQConnection** ()
- virtual void **removeSession** (**ActiveMQSession** *session) throw (cms::CMSException)
Removes the session resources for the given session instance.
- virtual void **addProducer** (**ActiveMQProducer** *producer) throw (cms::CMSException)
Adds an active Producer to the Set of known producers.
- virtual void **removeProducer** (const **Pointer**< **commands::ProducerId** > &producerId) throw (cms::CMSException)
Removes an active Producer to the Set of known producers.
- virtual void **addDispatcher** (const **Pointer**< **commands::ConsumerId** > &consumer, **Dispatcher** *dispatcher) throw (cms::CMSException)
Adds a dispatcher for a consumer.
- virtual void **removeDispatcher** (const **Pointer**< **commands::ConsumerId** > &consumer) throw (cms::CMSException)
Removes the dispatcher for a consumer.
- virtual void **sendPullRequest** (const **commands::ConsumerInfo** *consumer, long long timeout) throw (exceptions::ActiveMQException)
If supported sends a message pull request to the service provider asking for the delivery of a new message.

- bool **isClosed** () const
Checks if this connection has been closed.
- bool **isStarted** () const
Check if this connection has been started.
- virtual void **destroyDestination** (const **commands::ActiveMQDestination** *destination) throw (**decaf::lang::exceptions::NullPointerException**, **decaf::lang::exceptions::IllegalStateException**, **decaf::lang::exceptions::UnsupportedOperationException**, **activemq::exceptions::ActiveMQException**)
Requests that the Broker removes the given Destination.
- virtual void **destroyDestination** (const **cms::Destination** *destination) throw (**decaf::lang::exceptions::NullPointerException**, **decaf::lang::exceptions::IllegalStateException**, **decaf::lang::exceptions::UnsupportedOperationException**, **activemq::exceptions::ActiveMQException**)
Requests that the Broker removes the given Destination.
- virtual const **cms::ConnectionMetaData** * **getMetaData** () const throw (**cms::CMSEException**)
Gets the metadata for this connection.
- virtual **cms::Session** * **createSession** () throw (**cms::CMSEException**)
Creates a new Session to work for this Connection.
- virtual **cms::Session** * **createSession** (**cms::Session::AcknowledgeMode** ackMode) throw (**cms::CMSEException**)
Creates a new Session to work for this Connection using the specified acknowledgment mode.
- virtual std::string **getClientID** () const
Get the Client Id for this session.
- virtual void **close** () throw (**cms::CMSEException**)
Closes this connection as well as any Sessions created from it (and those Sessions' consumers and producers).
- virtual void **start** () throw (**cms::CMSEException**)
Starts or (restarts) a connections delivery of incoming messages.
- virtual void **stop** () throw (**cms::CMSEException**)
Stop the flow of incoming messages.
- virtual **cms::ExceptionListener** * **getExceptionListener** () const
Gets the registered Exception Listener for this connection.
- virtual void **setExceptionListener** (**cms::ExceptionListener** *listener)
Sets the registered Exception Listener for this connection.
- void **addTransportListener** (**transport::TransportListener** *transportListener)

Adds a transport listener so that a client can be notified of events in the underlying transport, client's are always notified after the event has been processed by the Connection class.

- **void removeTransportListener (transport::TransportListener *transportListener)**
Removes a registered TransportListener from the Connection's set of Transport listeners, this listener will no longer receive any Transport related events.
- **virtual void onCommand (const Pointer< commands::Command > &command)**
Event handler for the receipt of a non-response command from the transport.
- **virtual void onException (const decaf::lang::Exception &ex)**
Event handler for an exception from a command transport.
- **virtual void transportInterrupted ()**
The transport has suffered an interruption from which it hopes to recover.
- **virtual void transportResumed ()**
The transport has resumed after an interruption.
- **const commands::ConnectionInfo & getConnectionInfo () const throw (exceptions::ActiveMQException)**
Gets the ConnectionInfo for this Object, if the Connection is not open than this method throws an exception.
- **const commands::ConnectionId & getConnectionId () const throw (exceptions::ActiveMQException)**
Gets the ConnectionId for this Object, if the Connection is not open than this method throws an exception.
- **void oneway (Pointer< commands::Command > command) throw (activemq::exceptions::ActiveMQException)**
Sends a oneway message.
- **void syncRequest (Pointer< commands::Command > command, unsigned int timeout=0) throw (activemq::exceptions::ActiveMQException)**
Sends a synchronous request and returns the response from the broker.
- **virtual void fire (const exceptions::ActiveMQException &ex)**
Notify the exception listener.

6.21.1 Detailed Description

Concrete connection used for all connectors to the ActiveMQ broker.

6.21.2 Constructor & Destructor Documentation

- 6.21.2.1 activemq::core::ActiveMQConnection::ActiveMQConnection (const Pointer< transport::Transport > & transport, const Pointer< decaf::util::Properties > & properties)**

Constructor.

Parameters

transport The Transport requested for this connection to the Broker.

properties The Properties that were defined for this connection

6.21.2.2 `virtual activemq::core::ActiveMQConnection::~~ActiveMQConnection ()`
[virtual]

6.21.3 Member Function Documentation

6.21.3.1 `virtual void activemq::core::ActiveMQConnection::addDispatcher (`
`const Pointer< commands::ConsumerId > & consumer, Dispatcher *`
`dispatcher) throw (cms::CMSEException)` [virtual]

Adds a dispatcher for a consumer.

Parameters

consumer - The consumer for which to register a dispatcher.

dispatcher - The dispatcher to handle incoming messages for the consumer.

6.21.3.2 `virtual void activemq::core::ActiveMQConnection::addProducer (`
`ActiveMQProducer * producer) throw (cms::CMSEException)`
[virtual]

Adds an active Producer to the Set of known producers.

Parameters

producer - The Producer to add from the the known set.

6.21.3.3 `void activemq::core::ActiveMQConnection::addTransportListener (`
`transport::TransportListener * transportListener)`

Adds a transport listener so that a client can be notified of events in the underlying transport, client's are always notified after the event has been processed by the Connection class.

Client's should ensure that the registered listener does not block or take a long amount of time to execute in order to not degrade performance of this Connection.

Parameters

transportListener The TransportListener instance to add to this Connection's set of listeners to notify of Transport events.

6.21.3.4 `virtual void activemq::core::ActiveMQConnection::close () throw (`
`cms::CMSEException)` [virtual]

Closes this connection as well as any Sessions created from it (and those Sessions' consumers and producers).

Exceptions*CMSException*

6.21.3.5 `virtual cms::Session* activemq::core::ActiveMQConnection::createSession (cms::Session::AcknowledgeMode ackMode) throw (cms::CMSException)` [virtual]

Creates a new Session to work for this Connection using the specified acknowledgment mode.

Parameters

ackMode the Acknowledgment Mode to use.

Exceptions*CMSException*

6.21.3.6 `virtual cms::Session* activemq::core::ActiveMQConnection::createSession () throw (cms::CMSException)` [virtual]

Creates a new Session to work for this Connection.

Exceptions*CMSException*

6.21.3.7 `virtual void activemq::core::ActiveMQConnection::destroyDestination (const commands::ActiveMQDestination * destination) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalStateException, decaf::lang::exceptions::UnsupportedOperationException, activemq::exceptions::ActiveMQException)` [virtual]

Requests that the Broker removes the given Destination.

Calling this method implies that the client is finished with the Destination and that no other messages will be sent or received for the given Destination. The Broker frees all resources it has associated with this Destination.

Parameters

destination The Destination the Broker will be requested to remove.

Exceptions

NullPointerException If the passed Destination is Null

IllegalStateException If the connection is closed.

UnsupportedOperationException If the wire format in use does not support this operation.

ActiveMQException If any other error occurs during the attempt to destroy the destination.

6.21.3.8 `virtual void activemq::core::ActiveMQConnection::destroyDestination (const cms::Destination * destination) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalStateException, decaf::lang::exceptions::UnsupportedOperationException, activemq::exceptions::ActiveMQException) [virtual]`

Requests that the Broker removes the given Destination.

Calling this method implies that the client is finished with the Destination and that no other messages will be sent or received for the given Destination. The Broker frees all resources it has associated with this Destination.

Parameters

destination The CMS Destination the Broker will be requested to remove.

Exceptions

NullPointerException If the passed Destination is Null

IllegalStateException If the connection is closed.

UnsupportedOperationException If the wire format in use does not support this operation.

ActiveMQException If any other error occurs during the attempt to destroy the destination.

6.21.3.9 `virtual void activemq::core::ActiveMQConnection::fire (const exceptions::ActiveMQException & ex) [virtual]`

Notify the exception listener.

Parameters

ex the exception to fire

6.21.3.10 `virtual std::string activemq::core::ActiveMQConnection::getClientID () const [virtual]`

Get the Client Id for this session.

Returns

string version of Client Id

6.21.3.11 `const commands::ConnectionId& activemq::core::ActiveMQConnection::getConnectionId () const throw (exceptions::ActiveMQException)`

Gets the ConnectionId for this Object, if the Connection is not open than this method throws an exception.

6.21.3.12 `const commands::ConnectionInfo& activemq::core::ActiveMQConnection::getConnectionInfo () const throw (exceptions::ActiveMQException)`

Gets the ConnectionInfo for this Object, if the Connection is not open than this method throws an exception.

6.21.3.13 `virtual cms::ExceptionListener* activemq::core::ActiveMQConnection::getExceptionListener () const [inline, virtual]`

Gets the registered Exception Listener for this connection.

Returns

pointer to an exception listener or NULL

6.21.3.14 `virtual const cms::ConnectionMetaData* activemq::core::ActiveMQConnection::getMetaData () const throw (cms::CMSException) [inline, virtual]`

Gets the metadata for this connection.

Returns

the connection MetaData pointer (caller does not own it).

Exceptions

CMSException if the provider fails to get the connection metadata for this connection.

See also

ConnectionMetaData

Since

2.0

6.21.3.15 `bool activemq::core::ActiveMQConnection::isClosed () const [inline]`

Checks if this connection has been closed.

Returns

true if the connection is closed

6.21.3.16 `bool activemq::core::ActiveMQConnection::isStarted () const [inline]`

Check if this connection has been started.

Returns

true if the start method has been called.

6.21.3.17 virtual void activemq::core::ActiveMQConnection::onCommand (const Pointer< commands::Command > & *command*) [virtual]

Event handler for the receipt of a non-response command from the transport.

Parameters

command the received command object.

Implements **activemq::transport::TransportListener** (p. 3081).

6.21.3.18 void activemq::core::ActiveMQConnection::oneway (Pointer< commands::Command > *command*) throw (activemq::exceptions::ActiveMQException)

Sends a oneway message.

Parameters

command The message to send.

Exceptions

ConnectorException if not currently connected, or if the operation fails for any reason.

6.21.3.19 virtual void activemq::core::ActiveMQConnection::onException (const decaf::lang::Exception & *ex*) [virtual]

Event handler for an exception from a command transport.

Parameters

ex The exception.

Implements **activemq::transport::TransportListener** (p. 3082).

6.21.3.20 virtual void activemq::core::ActiveMQConnection::removeDispatcher (const Pointer< commands::ConsumerId > & *consumer*) throw (cms::CMSException) [virtual]

Removes the dispatcher for a consumer.

Parameters

consumer - The consumer for which to remove the dispatcher.

6.21.3.21 virtual void activemq::core::ActiveMQConnection::removeProducer (const Pointer< commands::ProducerId > & *producerId*) throw (cms::CMSException) [virtual]

Removes an active Producer to the Set of known producers.

Parameters

producerId - The ProducerId to remove from the the known set.

6.21.3.22 `virtual void activemq::core::ActiveMQConnection::removeSession (ActiveMQSession * session) throw (cms::CMSException) [virtual]`

Removes the session resources for the given session instance.

Parameters

session The session to be unregistered from this connection.

6.21.3.23 `void activemq::core::ActiveMQConnection::removeTransportListener (transport::TransportListener * transportListener)`

Removes a registered TransportListener from the Connection's set of Transport listeners, this listener will no longer receive any Transport related events.

The caller is responsible for freeing the listener in all cases.

Parameters

transportListener The pointer to the TransportListener to remove from the set of listeners.

6.21.3.24 `virtual void activemq::core::ActiveMQConnection::sendPullRequest (const commands::ConsumerInfo * consumer, long long timeout) throw (exceptions::ActiveMQException) [virtual]`

If supported sends a message pull request to the service provider asking for the delivery of a new message.

This is used in the case where the service provider has been configured with a zero prefetch or is only capable of delivering messages on a pull basis.

Parameters

consumer - the ConsumerInfo for the requesting Consumer.

timeout - the time that the client is willing to wait.

6.21.3.25 `virtual void activemq::core::ActiveMQConnection::setExceptionListener (cms::ExceptionListener * listener) [inline, virtual]`

Sets the registered Exception Listener for this connection.

Parameters

listener pointer to and ExceptionListener

6.21.3.26 `virtual void activemq::core::ActiveMQConnection::start () throw (cms::CMSException) [virtual]`

Starts or (restarts) a connections delivery of incoming messages.

Exceptions

CMSException

6.21.3.27 `virtual void activemq::core::ActiveMQConnection::stop () throw (cms::CMSException) [virtual]`

Stop the flow of incoming messages.

Exceptions

CMSException

6.21.3.28 `void activemq::core::ActiveMQConnection::syncRequest (Pointer< commands::Command > command, unsigned int timeout = 0) throw (activemq::exceptions::ActiveMQException)`

Sends a synchronous request and returns the response from the broker.

Converts any error responses into an exception.

Parameters

command The request command.

timeout The time to wait for a response, default is zero or infinite.

Exceptions

ConnectorException thrown if an error response was received from the broker, or if any other error occurred.

6.21.3.29 `virtual void activemq::core::ActiveMQConnection::transportInterrupted () [virtual]`

The transport has suffered an interruption from which it hopes to recover.

Implements `activemq::transport::TransportListener` (p. 3082).

6.21.3.30 `virtual void activemq::core::ActiveMQConnection::transportResumed () [virtual]`

The transport has resumed after an interruption.

Implements `activemq::transport::TransportListener` (p. 3082).

The documentation for this class was generated from the following file:

- `src/main/activemq/core/ActiveMQConnectionFactory.h`

6.22 activemq::core::ActiveMQConnectionFactory Class Reference

```
#include <src/main/activemq/core/ActiveMQConnectionFactory.h>
```

Inheritance diagram for `activemq::core::ActiveMQConnectionFactory`:

Public Member Functions

- **ActiveMQConnectionFactory** ()
- **ActiveMQConnectionFactory** (const std::string &url, const std::string &username="", const std::string &password="")
Constructor.
- virtual ~**ActiveMQConnectionFactory** ()
- virtual **cms::Connection** * **createConnection** () throw (cms::CMSException)
Creates a connection with the default user identity.
- virtual **cms::Connection** * **createConnection** (const std::string &username, const std::string &password) throw (cms::CMSException)
Creates a connection with the specified user identity.
- virtual **cms::Connection** * **createConnection** (const std::string &username, const std::string &password, const std::string &clientId) throw (cms::CMSException)
Creates a connection with the specified user identity.
- virtual void **setUsername** (const std::string &username)
Sets the username that should be used when creating a new connection.
- virtual const std::string & **getUsername** () const
Gets the username that this factory will use when creating a new connection instance.
- virtual void **setPassword** (const std::string &password)
Sets the password that should be used when creating a new connection.
- virtual const std::string & **getPassword** () const
Gets the password that this factory will use when creating a new connection instance.
- virtual void **setBrokerURL** (const std::string &brokerURL)
Sets the Broker URL that should be used when creating a new connection instance.
- virtual const std::string & **getBrokerURL** () const
Gets the Broker URL that this factory will use when creating a new connection instance.

Static Public Member Functions

- static **cms::Connection** * **createConnection** (const std::string &url, const std::string &username, const std::string &password, const std::string &clientId="") throw (cms::CMSException)
Creates a connection with the specified user identity.

6.22.1 Constructor & Destructor Documentation

6.22.1.1 `activemq::core::ActiveMQConnectionFactory::ActiveMQConnectionFactory ()`

6.22.1.2 `activemq::core::ActiveMQConnectionFactory::ActiveMQConnectionFactory (const std::string & url, const std::string & username = "", const std::string & password = "")`

Constructor.

Parameters

url the URL of the Broker we are connecting to.

username to authenticate with, defaults to ""

password to authenticate with, defaults to ""

6.22.1.3 `virtual activemq::core::ActiveMQConnectionFactory::~~ActiveMQConnectionFactory () [inline, virtual]`

6.22.2 Member Function Documentation

6.22.2.1 `virtual cms::Connection* activemq::core::ActiveMQConnectionFactory::createConnection () throw (cms::CMSEException) [virtual]`

Creates a connection with the default user identity.

The connection is created in stopped mode. No messages will be delivered until the Connection.start method is explicitly called.

Returns

a Connection Pointer

Exceptions

CMSEException

Implements `cms::ConnectionFactory` (p. 1105).

6.22.2.2 `virtual cms::Connection* activemq::core::ActiveMQConnectionFactory::createConnection (const std::string & username, const std::string & password) throw (cms::CMSEException) [virtual]`

Creates a connection with the specified user identity.

The connection is created in stopped mode. No messages will be delivered until the Connection.start method is explicitly called. The username and password values passed here do not change the defaults, subsequent calls to the parameterless createConnection will continue to use the default values that were set in the Constructor.

Parameters

username to authenticate with

password to authenticate with

Returns

a Connection Pointer

Exceptions

CMSEException

Implements **cms::ConnectionFactory** (p. 1104).

```
6.22.2.3 static cms::Connection* ac-
tivemq::core::ActiveMQConnectionFactory::createConnection (
    const std::string & url, const std::string & username, const
    std::string & password, const std::string & clientId = "" ) throw (
    cms::CMSEException ) [static]
```

Creates a connection with the specified user identity.

The connection is created in stopped mode. No messages will be delivered until the Connection.start method is explicitly called.

Parameters

url the URL of the Broker we are connecting to.

username to authenticate with

password to authenticate with

clientId to assign to connection, defaults to ""

Exceptions

CMSEException.

```
6.22.2.4 virtual cms::Connection* ac-
tivemq::core::ActiveMQConnectionFactory::createConnection ( const
    std::string & username, const std::string & password, const std::string
    & clientId ) throw ( cms::CMSEException ) [virtual]
```

Creates a connection with the specified user identity.

The connection is created in stopped mode. No messages will be delivered until the Connection.start method is explicitly called. The username and password values passed here do not change the defaults, subsequent calls to the parameterless createConnection will continue to use the default values that were set in the Constructor.

Parameters

username to authenticate with

password to authenticate with

clientId to assign to connection if "" then a random cleint Id is created for this connection.

Returns

a Connection Pointer

Exceptions

CMSException

Implements **cms::ConnectionFactory** (p. 1105).

6.22.2.5 `virtual const std::string& activemq::core::ActiveMQConnectionFactory::getBrokerURL () const [inline, virtual]`

Gets the Broker URL that this factory will use when creating a new connection instance.

Returns

brokerURL string

6.22.2.6 `virtual const std::string& activemq::core::ActiveMQConnectionFactory::getPassword () const [inline, virtual]`

Gets the password that this factory will use when creating a new connection instance.

Returns

password string, "" for default credentials

6.22.2.7 `virtual const std::string& activemq::core::ActiveMQConnectionFactory::getUsername () const [inline, virtual]`

Gets the username that this factory will use when creating a new connection instance.

Returns

username string, "" for default credentials

6.22.2.8 `virtual void activemq::core::ActiveMQConnectionFactory::setBrokerURL (const std::string & brokerURL) [inline, virtual]`

Sets the Broker URL that should be used when creating a new connection instance.

Parameters

brokerURL string

6.22.2.9 virtual void activemq::core::ActiveMQConnectionFactory::setPassword (const std::string & *password*) [inline, virtual]

Sets the password that should be used when creating a new connection.

Parameters

password string

6.22.2.10 virtual void activemq::core::ActiveMQConnectionFactory::setUsername (const std::string & *username*) [inline, virtual]

Sets the username that should be used when creating a new connection.

Parameters

username string

The documentation for this class was generated from the following file:

- src/main/activemq/core/ActiveMQConnectionFactory.h

6.23 activemq::core::ActiveMQConnectionMetaData Class Reference

This class houses all the various settings and information that is used by an instance of an **ActiveMQConnection** (p. 202) class.

```
#include <src/main/activemq/core/ActiveMQConnectionMetaData.h>
```

Inheritance diagram for activemq::core::ActiveMQConnectionMetaData:

Public Member Functions

- **ActiveMQConnectionMetaData** ()
- virtual **~ActiveMQConnectionMetaData** ()
- virtual std::string **getCMSVersion** () const throw (cms::CMSException)
Gets the CMS API version.
- virtual int **getCMSMajorVersion** () const throw (cms::CMSException)
Gets the CMS major version number.
- virtual int **getCMSMinorVersion** () const throw (cms::CMSException)
Gets the CMS minor version number.
- virtual std::string **getCMSProviderName** () const throw (cms::CMSException)
Gets the CMS provider name.

- virtual std::string **getProviderVersion** () const throw (cms::CMSException)
Gets the CMS provider version.
- virtual int **getProviderMajorVersion** () const throw (cms::CMSException)
Gets the CMS provider major version number.
- virtual int **getProviderMinorVersion** () const throw (cms::CMSException)
Gets the CMS provider minor version number.
- virtual std::vector< std::string > **getCMSXPropertyNames** () const throw (cms::CMSException)
Gets an Vector of the CMSX property names.

6.23.1 Detailed Description

This class houses all the various settings and information that is used by an instance of an **ActiveMQConnection** (p. 202) class.

Since

3.0

6.23.2 Constructor & Destructor Documentation

6.23.2.1 **activemq::core::ActiveMQConnectionMetaData::ActiveMQConnectionMetaData** ()

6.23.2.2 **virtual**
activemq::core::ActiveMQConnectionMetaData::~~ActiveMQConnectionMetaData () [virtual]

6.23.3 Member Function Documentation

6.23.3.1 **virtual int** **activemq::core::ActiveMQConnectionMetaData::getCMSMajorVersion** ()
const throw (cms::CMSException) [virtual]

Gets the CMS major version number.

Returns

the CMS API major version number

Exceptions

CMSException If the CMS Provider fails to retrieve the metadata due to some internal error.

Implements **cms::ConnectionMetaData** (p. 1155).

6.23.3.2 `virtual int activemq::core::ActiveMQConnectionMetaData::getCMSMinorVersion () const throw (cms::CMSEException) [virtual]`

Gets the CMS minor version number.

Returns

the CMS API minor version number

Exceptions

CMSEException If the CMS Provider fails to retrieve the metadata due to some internal error.

Implements `cms::ConnectionMetaData` (p.1156).

6.23.3.3 `virtual std::string activemq::core::ActiveMQConnectionMetaData::getCMSProviderName () const throw (cms::CMSEException) [virtual]`

Gets the CMS provider name.

Returns

the CMS provider name

Exceptions

CMSEException If the CMS Provider fails to retrieve the metadata due to some internal error.

Implements `cms::ConnectionMetaData` (p.1156).

6.23.3.4 `virtual std::string activemq::core::ActiveMQConnectionMetaData::getCMSVersion () const throw (cms::CMSEException) [virtual]`

Gets the CMS API version.

Returns

the CMS API Version in String form.

Exceptions

CMSEException If the CMS Provider fails to retrieve the metadata due to some internal error.

Implements `cms::ConnectionMetaData` (p.1156).

6.23.3.5 `virtual std::vector<std::string> activemq::core::ActiveMQConnectionMetaData::getCMSXPropertyNames () const throw (cms::CMSException) [virtual]`

Gets an Vector of the CMSX property names.

Returns

an Vector of CMSX property names

Exceptions

CMSException If the CMS Provider fails to retrieve the metadata due to some internal error.

Implements `cms::ConnectionMetaData` (p.1157).

6.23.3.6 `virtual int activemq::core::ActiveMQConnectionMetaData::getProviderMajorVersion () const throw (cms::CMSException) [virtual]`

Gets the CMS provider major version number.

Returns

the CMS provider major version number

Exceptions

CMSException If the CMS Provider fails to retrieve the metadata due to some internal error.

Implements `cms::ConnectionMetaData` (p.1157).

6.23.3.7 `virtual int activemq::core::ActiveMQConnectionMetaData::getProviderMinorVersion () const throw (cms::CMSException) [virtual]`

Gets the CMS provider minor version number.

Returns

the CMS provider minor version number

Exceptions

CMSException If the CMS Provider fails to retrieve the metadata due to some internal error.

Implements `cms::ConnectionMetaData` (p.1157).

6.23.3.8 `virtual std::string activemq::core::ActiveMQConnectionMetaData::getProviderVersion () const throw (cms::CMSException) [virtual]`

Gets the CMS provider version.

Returns

the CMS provider version

Exceptions

CMSException If the CMS Provider fails to retrieve the metadata due to some internal error.

Implements `cms::ConnectionMetaData` (p.1157).

The documentation for this class was generated from the following file:

- `src/main/activemq/core/ActiveMQConnectionMetaData.h`

6.24 `activemq::core::ActiveMQConnectionSupport` Class Reference

`#include <src/main/activemq/core/ActiveMQConnectionSupport.h>`

Inheritance diagram for `activemq::core::ActiveMQConnectionSupport`:

Public Member Functions

- `ActiveMQConnectionSupport (const Pointer< transport::Transport > &transport, const Pointer< decaf::util::Properties > &properties)`
*Creates an instance of the **ActiveMQConnectionSupport** (p.220) class, the most common properties for a connection are pulled from the properties instance or are set to defaults.*
- `virtual ~ActiveMQConnectionSupport ()`
- `virtual void startupTransport () throw (decaf::lang::Exception)`
Starts the Transport, this should initiate the connection between this client and the Transports endpoint.
- `virtual void shutdownTransport () throw (decaf::lang::Exception)`
Closes this object and deallocates the appropriate resources.
- `const decaf::util::Properties & getProperties () const`
Gets the Properties object that this Config object was initialized with.
- `transport::Transport & getTransport () const`
Gets the Transport Configured for this Connection.

- **bool isAlwaysSyncSend () const**
Gets if the Connection should always send things Synchronously.
- **void setAlwaysSyncSend (bool value)**
Sets if the Connection should always send things Synchronously.
- **bool isUseAsyncSend () const**
Gets if the useAsyncSend option is set.
- **void setUseAsyncSend (bool value)**
Sets the useAsyncSend option.
- **unsigned int getSendTimeout () const**
Gets the assigned send timeout for this Connector.
- **void setSendTimeout (unsigned int timeout)**
Sets the send timeout to use when sending Message objects, this will cause all messages to be sent using a Synchronous request is non-zero.
- **unsigned int getCloseTimeout () const**
Gets the assigned close timeout for this Connector.
- **void setCloseTimeout (unsigned int timeout)**
Sets the close timeout to use when sending the disconnect request.
- **unsigned int getProducerWindowSize () const**
Gets the configured producer window size for Producers that are created from this connector.
- **void setProducerWindowSize (unsigned int windowSize)**
Sets the size in Bytes of messages that a producer can send before it is blocked to await a ProducerAck from the broker that frees enough memory to allow another message to be sent.
- **std::string getUsername () const**
Gets the Configured Username.
- **void setUsername (const std::string &username)**
Sets the Username.
- **std::string getPassword () const**
Gets the Configured Password.
- **void setPassword (const std::string &password)**
Sets the Password.
- **std::string getClientId () const**
Gets the Configured Client Id.
- **void setClientId (const std::string &clientId)**
Sets the Client Id.

- long long **getNextSessionId** ()
Get the Next available Session Id.
- long long **getNextTempDestinationId** ()
Get the Next Temporary Destination Id.
- long long **getNextLocalTransactionId** ()
Get the Next Temporary Destination Id.

6.24.1 Constructor & Destructor Documentation

6.24.1.1 **activemq::core::ActiveMQConnectionSupport::ActiveMQConnectionSupport** (const Pointer< transport::Transport > & *transport*, const Pointer< decaf::util::Properties > & *properties*)

Creates an instance of the **ActiveMQConnectionSupport** (p.220) class, the most common properties for a connection are pulled from the properties instance or are set to defaults.

Parameters

transport The Transport that this Connection will use for sending Commands to the Broker.
properties The URI configured properties for this connection.

6.24.1.2 **virtual** **activemq::core::ActiveMQConnectionSupport::~~ActiveMQConnectionSupport** () [virtual]

6.24.2 Member Function Documentation

6.24.2.1 **std::string activemq::core::ActiveMQConnectionSupport::getClientId** () **const** [inline]

Gets the Configured Client Id.

Returns

the clientId.

6.24.2.2 **unsigned int ac-** **tivemq::core::ActiveMQConnectionSupport::getCloseTimeout** () **const** [inline]

Gets the assigned close timeout for this Connector.

Returns

the close timeout configured in the connection uri

6.24.2.3 `long long activemq::core::ActiveMQConnectionSupport::getNextLocalTransactionId () [inline]`

Get the Next Temporary Destination Id.

Returns

the next id in the sequence.

6.24.2.4 `long long activemq::core::ActiveMQConnectionSupport::getNextSessionId () [inline]`

Get the Next available Session Id.

Returns

the next id in the sequence.

6.24.2.5 `long long activemq::core::ActiveMQConnectionSupport::getNextTempDestinationId () [inline]`

Get the Next Temporary Destination Id.

Returns

the next id in the sequence.

6.24.2.6 `std::string activemq::core::ActiveMQConnectionSupport::getPassword () const [inline]`

Gets the Configured Password.

Returns

the password.

6.24.2.7 `unsigned int activemq::core::ActiveMQConnectionSupport::getProducerWindowSize () const [inline]`

Gets the configured producer window size for Producers that are created from this connector.

This only applies if there is no send timeout and the producer is able to send asynchronously.

Returns

size in bytes of messages that this producer can produce before it must block and wait for ProducerAck messages to free resources.

6.24.2.8 `const decaf::util::Properties& activemq::core::ActiveMQConnectionSupport::getProperties () const [inline]`

Gets the Properties object that this Config object was initialized with.

Returns

a const reference to the Connection Config.

References `decaf::lang::Pointer< T, REFCOUNTER >::get()`.

6.24.2.9 `unsigned int activemq::core::ActiveMQConnectionSupport::getSendTimeout () const [inline]`

Gets the assigned send timeout for this Connector.

Returns

the send timeout configured in the connection uri

6.24.2.10 `transport::Transport& activemq::core::ActiveMQConnectionSupport::getTransport () const [inline]`

Gets the Transport Configured for this Connection.

Returns

the configured transport

References `decaf::lang::Pointer< T, REFCOUNTER >::get()`.

6.24.2.11 `std::string activemq::core::ActiveMQConnectionSupport::getUsername () const [inline]`

Gets the Configured Username.

Returns

the username.

6.24.2.12 `bool activemq::core::ActiveMQConnectionSupport::isAlwaysSyncSend () const [inline]`

Gets if the Connection should always send things Synchronously.

Returns

true if sends should always be Synchronous.

6.24.2.13 `bool activemq::core::ActiveMQConnectionSupport::isUseAsyncSend ()`
`const [inline]`

Gets if the useAsyncSend option is set.

Returns

true if on false if not.

6.24.2.14 `void activemq::core::ActiveMQConnectionSupport::setAlwaysSyncSend (`
`bool value) [inline]`

Sets if the Connection should always send things Synchronously.

Parameters

value true if sends should always be Synchronous.

6.24.2.15 `void activemq::core::ActiveMQConnectionSupport::setClientId (const`
`std::string & clientId) [inline]`

Sets the Client Id.

Parameters

clientId - The new clientId value.

6.24.2.16 `void activemq::core::ActiveMQConnectionSupport::setCloseTimeout (`
`unsigned int timeout) [inline]`

Sets the close timeout to use when sending the disconnect request.

Parameters

timeout - The time to wait for a close message.

6.24.2.17 `void activemq::core::ActiveMQConnectionSupport::setPassword (const`
`std::string & password) [inline]`

Sets the Password.

Parameters

password - The new password value.

6.24.2.18 `void activemq::core::ActiveMQConnectionSupport::setProducerWindowSize (unsigned int windowSize) [inline]`

Sets the size in Bytes of messages that a producer can send before it is blocked to await a ProducerAck from the broker that frees enough memory to allow another message to be sent.

Parameters

windowSize - The size in bytes of the Producers memory window.

6.24.2.19 `void activemq::core::ActiveMQConnectionSupport::setSendTimeout (unsigned int timeout) [inline]`

Sets the send timeout to use when sending Message objects, this will cause all messages to be sent using a Synchronous request is non-zero.

Parameters

timeout - The time to wait for a response.

6.24.2.20 `void activemq::core::ActiveMQConnectionSupport::setUseAsyncSend (bool value) [inline]`

Sets the useAsyncSend option.

Parameters

value - true to activate, false to disable.

6.24.2.21 `void activemq::core::ActiveMQConnectionSupport::setUsername (const std::string & username) [inline]`

Sets the Username.

Parameters

username - The new username value.

6.24.2.22 `virtual void activemq::core::ActiveMQConnectionSupport::shutdownTransport () throw (decaf::lang::Exception) [virtual]`

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling close.

Exceptions

Exception

```
6.24.2.23 virtual void ac-
        tivemq::core::ActiveMQConnectionSupport::startupTransport (    )
        throw ( decaf::lang::Exception ) [virtual]
```

Starts the Transport, this should initiate the connection between this client and the Transports endpoint.

Exceptions

Exception

The documentation for this class was generated from the following file:

- src/main/activemq/core/ActiveMQConnectionSupport.h

6.25 activemq::core::ActiveMQConstants Class Reference

Class holding constant values for various ActiveMQ specific things Each constant is defined as an enumeration and has functions that convert back and forth between string and enum values.

```
#include <src/main/activemq/core/ActiveMQConstants.h>
```

Data Structures

- class StaticInitializer

Public Types

- enum TransactionState {
TRANSACTION_STATE_BEGIN = 0, TRANSACTION_STATE_PREPARE = 1, TRANSACTION_STATE_COMMITONEPHASE = 2, TRANSACTION_STATE_COMMITTWOPHASE = 3, TRANSACTION_STATE_ROLLBACK = 4, TRANSACTION_STATE_RECOVER = 5, TRANSACTION_STATE_FORGET = 6, TRANSACTION_STATE_END = 7 }
- enum DestinationActions { DESTINATION_ADD_OPERATION = 0, DESTINATION_REMOVE_OPERATION = 1 }
- enum AckType {
ACK_TYPE_DELIVERED = 0, ACK_TYPE_POISON = 1, ACK_TYPE_CONSUMED = 2, ACK_TYPE_REDELIVERED = 3, ACK_TYPE_INDIVIDUAL = 4 }
- enum DestinationOption {
CONSUMER_PREFETCHSIZE, CUNSUMER_MAXPENDINGMSGLIMIT, CONSUMER_NOLOCAL, CONSUMER_DISPATCHASYNC, CONSUMER_RETROACTIVE, CONSUMER_SELECTOR, CONSUMER_EXCLUSIVE, CONSUMER_PRIORITY, NUM_OPTIONS }

These values represent the options that can be appended to an Destination name, i.e.

- enum **URIParam** {
CONNECTION_SENDTIMEOUT, **CONNECTION_PRODUCERWINDOWSIZE**, **CONNECTION_CLOSETIMEOUT**,
CONNECTION_ALWAYSSENDCSEND,
CONNECTION_USEASYNCSSEND, **PARAM_USERNAME**, **PARAM_PASSWORD**, **PARAM_CLIENTID**,
NUM_PARAMS }

These values represent the parameters that can be added to the connection URI that affect the ActiveMQ Core API.

Static Public Member Functions

- static const std::string & **toString** (const **DestinationOption** option)
- static **DestinationOption** **toDestinationOption** (const std::string &option)
- static const std::string & **toString** (const **URIParam** option)
- static **URIParam** **toURIOption** (const std::string &option)

6.25.1 Detailed Description

Class holding constant values for various ActiveMQ specific things Each constant is defined as an enumeration and has functions that convert back an forth between string and enum values.

6.25.2 Member Enumeration Documentation

6.25.2.1 enum activemq::core::ActiveMQConstants::AckType

Enumerator:

ACK_TYPE_DELIVERED
ACK_TYPE_POISON
ACK_TYPE_CONSUMED
ACK_TYPE_REDELIVERED
ACK_TYPE_INDIVIDUAL

6.25.2.2 enum activemq::core::ActiveMQConstants::DestinationActions

Enumerator:

DESTINATION_ADD_OPERATION
DESTINATION_REMOVE_OPERATION

6.25.2.3 enum activemq::core::ActiveMQConstants::DestinationOption

These values represent the options that can be appended to an Destination name, i.e.

/topic/foo?consumer.exclusive=true

Enumerator:

```
CONSUMER_PREFECTCHSIZE  
CUNSUMER_MAXPENDINGMSGLIMIT  
CONSUMER_NOLOCAL  
CONSUMER_DISPATCHASYNC  
CONSUMER_RETROACTIVE  
CONSUMER_SELECTOR  
CONSUMER_EXCLUSIVE  
CONSUMER_PRIORITY  
NUM_OPTIONS
```

6.25.2.4 enum activemq::core::ActiveMQConstants::TransactionState

Enumerator:

```
TRANSACTION_STATE_BEGIN  
TRANSACTION_STATE_PREPARE  
TRANSACTION_STATE_COMMITONEPHASE  
TRANSACTION_STATE_COMMITTWOPHASE  
TRANSACTION_STATE_ROLLBACK  
TRANSACTION_STATE_RECOVER  
TRANSACTION_STATE_FORGET  
TRANSACTION_STATE_END
```

6.25.2.5 enum activemq::core::ActiveMQConstants::URIPParam

These values represent the parameters that can be added to the connection URI that affect the ActiveMQ Core API.

Enumerator:

```
CONNECTION_SENDTIMEOUT  
CONNECTION_PRODUCERWINDOWSIZE  
CONNECTION_CLOSETIMEOUT  
CONNECTION_ALWAYSASYNCSEND  
CONNECTION_USEASYNCSEND  
PARAM_USERNAME  
PARAM_PASSWORD  
PARAM_CLIENTID  
NUM_PARAMS
```

6.25.3 Member Function Documentation

- 6.25.3.1 static `DestinationOption` `activemq::core::ActiveMQConstants::toDestinationOption (const std::string & option)` [inline, static]
- 6.25.3.2 static const std::string& `activemq::core::ActiveMQConstants::toString (const DestinationOption option)` [inline, static]
- 6.25.3.3 static const std::string& `activemq::core::ActiveMQConstants::toString (const URIParam option)` [inline, static]
- 6.25.3.4 static `URIParam` `activemq::core::ActiveMQConstants::toURIOption (const std::string & option)` [inline, static]

The documentation for this class was generated from the following file:

- `src/main/activemq/core/ActiveMQConstants.h`

6.26 activemq::core::ActiveMQConsumer Class Reference

```
#include <src/main/activemq/core/ActiveMQConsumer.h>
```

Inheritance diagram for `activemq::core::ActiveMQConsumer`:

Public Member Functions

- **ActiveMQConsumer** (const **Pointer**< **commands::ConsumerInfo** > &consumerInfo, **ActiveMQSession** *session, const **Pointer**< **ActiveMQTransactionContext** > &transaction)
Constructor.
- virtual **~ActiveMQConsumer** ()
- virtual void **start** ()
Starts the Consumer if not already started and not closed.
- virtual void **stop** ()
Stops a Consumer, the Consumer will not deliver any messages that are dispatched to it until it is started again.
- virtual void **close** () throw (cms::CMSException)
Closes the Consumer.
- virtual **cms::Message** * **receive** () throw (cms::CMSException)
Synchronously Receive a Message.
- virtual **cms::Message** * **receive** (int millisecs) throw (cms::CMSException)
Synchronously Receive a Message, time out after defined interval.

- virtual **cms::Message** * **receiveNoWait** () throw (cms::CMSEException)
Receive a Message, does not wait if there isn't a new message to read, returns NULL if nothing read.
- virtual void **setMessageListener** (cms::MessageListener *listener) throw (cms::CMSEException)
Sets the MessageListener that this class will send notifs on.
- virtual **cms::MessageListener** * **getMessageListener** () const throw (cms::CMSEException)
Gets the MessageListener that this class will send events to.
- virtual std::string **getMessageSelector** () const throw (cms::CMSEException)
Gets this message consumer's message selector expression.
- virtual void **acknowledge** (const Pointer< commands::MessageDispatch > &dispatch) throw (cms::CMSEException)
Method called to acknowledge the message passed, called from a message when the mode is client ack.
- virtual void **dispatch** (const Pointer< MessageDispatch > &message)
Called asynchronously by the session to dispatch a message.
- void **acknowledge** () throw (cms::CMSEException)
Method called to acknowledge all messages that have been received so far.
- void **commit** () throw (exceptions::ActiveMQException)
Called to Commit the current set of messages in this Transaction.
- void **rollback** () throw (exceptions::ActiveMQException)
Called to Roll back the current set of messages in this Transaction.
- void **doClose** () throw (exceptions::ActiveMQException)
Performs the actual close operation on this consumer.
- const **commands::ConsumerInfo** & **getConsumerInfo** () const
Get the Consumer information for this consumer.
- const **commands::ConsumerId** & **getConsumerId** () const
Get the Consumer Id for this consumer.
- bool **isClosed** () const
- bool **isSynchronizationRegistered** () const
*Has this Consumer Transaction **Synchronization** (p. 2945) been added to the transaction.*
- void **setSynchronizationRegistered** (bool value)
*Sets the **Synchronization** (p. 2945) Registered state of this consumer.*
- bool **iterate** ()

Deliver any pending messages to the registered MessageListener if there is one, return true if not all dispatched, or false if no listener or all pending messages have been dispatched.

- void **deliverAcks** () throw (exceptions::ActiveMQException)

Forces this consumer to send all pending acks to the broker.

- void **clearMessagesInProgress** ()

Called on a Failover to clear any pending messages.

- long long **getLastDeliveredSequenceId** () const

Gets the currently set Last Delivered Sequence Id.

- void **setLastDeliveredSequenceId** (long long value)

Sets the value of the Last Delivered Sequence Id.

Protected Member Functions

- **Pointer< MessageDispatch > dequeue** (long long timeout) throw (cms::CMSException)

Used by synchronous receive methods to wait for messages to come in.

- void **beforeMessageIsConsumed** (const **Pointer< commands::MessageDispatch >** &dispatch)

Pre-consume processing.

- void **afterMessageIsConsumed** (const **Pointer< commands::MessageDispatch >** &dispatch, bool messageExpired)

Post-consume processing.

6.26.1 Constructor & Destructor Documentation

- 6.26.1.1** **activemq::core::ActiveMQConsumer::ActiveMQConsumer** (const **Pointer< commands::ConsumerInfo >** & *consumerInfo*, **ActiveMQSession** * *session*, const **Pointer< ActiveMQTransactionContext >** & *transaction*)

Constructor.

- 6.26.1.2** **virtual activemq::core::ActiveMQConsumer::~~ActiveMQConsumer** ()
[virtual]

6.26.2 Member Function Documentation

- 6.26.2.1** **virtual void activemq::core::ActiveMQConsumer::acknowledge** (const **Pointer< commands::MessageDispatch >** & *dispatch*) throw (**cms::CMSException**) [virtual]

Method called to acknowledge the message passed, called from a message when the mode is client ack.

Parameters

message the Message to Acknowledge

Exceptions

CMSEException

6.26.2.2 void activemq::core::ActiveMQConsumer::acknowledge () throw (cms::CMSEException)

Method called to acknowledge all messages that have been received so far.

Exceptions

CMSEException

6.26.2.3 void activemq::core::ActiveMQConsumer::afterMessageIsConsumed (const Pointer< commands::MessageDispatch > & *dispatch*, bool *messageExpired*) [protected]

Post-consume processing.

Parameters

dispatch - the consumed message

messageExpired - flag indicating if the message has expired.

6.26.2.4 void activemq::core::ActiveMQConsumer::beforeMessageIsConsumed (const Pointer< commands::MessageDispatch > & *dispatch*) [protected]

Pre-consume processing.

Parameters

dispatch - the message being consumed.

6.26.2.5 void activemq::core::ActiveMQConsumer::clearMessagesInProgress ()

Called on a Failover to clear any pending messages.

6.26.2.6 virtual void activemq::core::ActiveMQConsumer::close () throw (cms::CMSEException) [virtual]

Closes the Consumer.

This will return all allocated resources and purge any outstanding messages. This method will block if there is a call to receive in progress, or a dispatch to a MessageListener in place

Exceptions

CMSEException

6.26.2.7 `void activemq::core::ActiveMQConsumer::commit () throw (exceptions::ActiveMQException)`

Called to Commit the current set of messages in this Transaction.

Exceptions

ActiveMQException

6.26.2.8 `void activemq::core::ActiveMQConsumer::deliverAcks () throw (exceptions::ActiveMQException)`

Forces this consumer to send all pending acks to the broker.

6.26.2.9 `Pointer<MessageDispatch> activemq::core::ActiveMQConsumer::dequeue (long long timeout) throw (cms::CMSException)` [protected]

Used by synchronous receive methods to wait for messages to come in.

Parameters

timeout - The maximum number of milliseconds to wait before returning. If -1, it will block until a messages is received or this consumer is closed. If 0, will not block at all. If > 0, will wait at a maximum the specified number of milliseconds before returning.

Returns

the message, if received within the allotted time. Otherwise NULL.

Exceptions

InvalidStateException if this consumer is closed upon entering this method.

6.26.2.10 `virtual void activemq::core::ActiveMQConsumer::dispatch (const Pointer< MessageDispatch > & message)` [virtual]

Called asynchronously by the session to dispatch a message.

Parameters

message dispatch object pointer

Implements `activemq::core::Dispatcher` (p. 1441).

6.26.2.11 `void activemq::core::ActiveMQConsumer::doClose () throw (exceptions::ActiveMQException)`

Performs the actual close operation on this consumer.

Exceptions

ActiveMQException

6.26.2.12 `const commands::ConsumerId& activemq::core::ActiveMQConsumer::getConsumerId ()
const [inline]`

Get the Consumer Id for this consumer.

Returns

Reference to a Consumer Id Object

6.26.2.13 `const commands::ConsumerInfo& activemq::core::ActiveMQConsumer::getConsumerInfo ()
const [inline]`

Get the Consumer information for this consumer.

Returns

Reference to a Consumer Info Object

6.26.2.14 `long long activemq::core::ActiveMQConsumer::getLastDeliveredSequenceId ()
const [inline]`

Gets the currently set Last Delivered Sequence Id.

Returns

long long containing the sequence id of the last delivered Message.

6.26.2.15 `virtual cms::MessageListener* activemq::core::ActiveMQConsumer::getMessageListener ()
const throw (cms::CMSEException) [inline, virtual]`

Gets the MessageListener that this class will send events to.

Returns

the currently registered MessageListener interface pointer.

6.26.2.16 `virtual std::string activemq::core::ActiveMQConsumer::getMessageSelector ()
const throw (cms::CMSEException) [virtual]`

Gets this message consumer's message selector expression.

Returns

This Consumer's selector expression or "".

Exceptions

cms::CMSEException (p. 960)

6.26.2.17 `bool activemq::core::ActiveMQConsumer::isClosed () const [inline]`**Returns**

if this Consumer has been closed.

6.26.2.18 `bool activemq::core::ActiveMQConsumer::isSynchronizationRegistered () const [inline]`

Has this Consumer Transaction **Synchronization** (p. 2945) been added to the transaction.

Returns

true if the synchronization has been added.

6.26.2.19 `bool activemq::core::ActiveMQConsumer::iterate ()`

Deliver any pending messages to the registered MessageListener if there is one, return true if not all dispatched, or false if no listener or all pending messages have been dispatched.

6.26.2.20 `virtual cms::Message* activemq::core::ActiveMQConsumer::receive () throw (cms::CMSException) [virtual]`

Synchronously Receive a Message.

Returns

new message

Exceptions

CMSException

6.26.2.21 `virtual cms::Message* activemq::core::ActiveMQConsumer::receive (int milliseconds) throw (cms::CMSException) [virtual]`

Synchronously Receive a Message, time out after defined interval.

Returns null if nothing read.

Parameters

milliseconds the time in milliseconds to wait before returning

Returns

new message or null on timeout

Exceptions

CMSException

6.26.2.22 `virtual cms::Message* activemq::core::ActiveMQConsumer::receiveNoWait () throw (cms::CMSException) [virtual]`

Receive a Message, does not wait if there isn't a new message to read, returns NULL if nothing read.

Returns

new message

Exceptions

CMSException

6.26.2.23 `void activemq::core::ActiveMQConsumer::rollback () throw (exceptions::ActiveMQException)`

Called to Roll back the current set of messages in this Transaction.

Exceptions

ActiveMQException

6.26.2.24 `void activemq::core::ActiveMQConsumer::setLastDeliveredSequenceId (long long value) [inline]`

Sets the value of the Last Delivered Sequence Id.

Parameters

value The new value to assign to the Last Delivered Sequence Id property.

6.26.2.25 `virtual void activemq::core::ActiveMQConsumer::setMessageListener (cms::MessageListener * listener) throw (cms::CMSException) [virtual]`

Sets the MessageListener that this class will send notifs on.

Parameters

listener MessageListener interface pointer

6.26.2.26 `void activemq::core::ActiveMQConsumer::setSynchronizationRegistered (bool value) [inline]`

Sets the **Synchronization** (p. 2945) Registered state of this consumer.

Parameters

value - true if registered false otherwise.

6.26.2.27 virtual void activemq::core::ActiveMQConsumer::start () [virtual]

Starts the Consumer if not already started and not closed.

A consumer will no deliver messages until started.

6.26.2.28 virtual void activemq::core::ActiveMQConsumer::stop () [virtual]

Stops a Consumer, the Consumer will not deliver any messages that are dispatched to it until it is started again.

A Closed Consumer is also a stopped consumer.

The documentation for this class was generated from the following file:

- `src/main/activemq/core/ActiveMQConsumer.h`

6.27 activemq::library::ActiveMQCPP Class Reference

```
#include <src/main/activemq/library/ActiveMQCPP.h>
```

Public Member Functions

- virtual `~ActiveMQCPP ()`

Static Public Member Functions

- static void `initializeLibrary ()`

Initialize the ActiveMQ-CPP Library constructs, this method will init all the internal Registry objects and initialize the Decaf library.

- static void `initializeLibrary (int argc, char **argv)`

Initialize the ActiveMQ-CPP Library constructs, this method will initialize all the internal Registry objects and initialize the Decaf library.

- static void `shutdownLibrary ()`

Shutdown the ActiveMQ-CPP Library, freeing any resources that could not be freed up to this point.

Protected Member Functions

- `ActiveMQCPP ()`
- `ActiveMQCPP (const ActiveMQCPP &)`
- `ActiveMQCPP & operator= (const ActiveMQCPP &)`

6.27.1 Constructor & Destructor Documentation

6.27.1.1 `activemq::library::ActiveMQCPP::ActiveMQCPP ()` [inline, protected]

6.27.1.2 `activemq::library::ActiveMQCPP::ActiveMQCPP (const ActiveMQCPP &)` [protected]

6.27.1.3 `virtual activemq::library::ActiveMQCPP::~~ActiveMQCPP ()` [inline, virtual]

6.27.2 Member Function Documentation

6.27.2.1 `static void activemq::library::ActiveMQCPP::initializeLibrary ()` [static]

Initialize the ActiveMQ-CPP Library constructs, this method will init all the internal Registry objects and initialize the Decaf library.

Exceptions

runtime_error if an error occurs while initializing this library.

6.27.2.2 `static void activemq::library::ActiveMQCPP::initializeLibrary (int argc, char ** argv)` [static]

Initialize the ActiveMQ-CPP Library constructs, this method will initialize all the internal Registry objects and initialize the Decaf library.

This method takes the args passed to the main method of process for use is setting system properties and configuring the ActiveMQ-CPP Library.

Parameters

argc - the count of arguments passed to this Process.

argv - the array of string arguments passed to this process.

Exceptions

runtime_error if an error occurs while initializing this library.

6.27.2.3 `ActiveMQCPP& activemq::library::ActiveMQCPP::operator= (const ActiveMQCPP &)` [protected]

6.27.2.4 `static void activemq::library::ActiveMQCPP::shutdownLibrary ()` [static]

Shutdown the ActiveMQ-CPP Library, freeing any resources that could not be freed up to this point.

All the user created ActiveMQ-CPP objects and Decaf Library objects should have been destroyed by the time this method is called.

The documentation for this class was generated from the following file:

- `src/main/activemq/library/ActiveMQCPP.h`

6.28 `activemq::commands::ActiveMQDestination` Class Reference

```
#include <src/main/activemq/commands/ActiveMQDestination.h>
```

Inheritance diagram for `activemq::commands::ActiveMQDestination`:

Data Structures

- struct `DestinationFilter`

Public Member Functions

- `ActiveMQDestination ()`
- `ActiveMQDestination (const std::string &physicalName)`
- virtual `~ActiveMQDestination ()`
- virtual `ActiveMQDestination * cloneDataStructure () const`
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void `copyDataStructure (const DataStructure *src)`
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual bool `equals (const DataStructure *value) const`
*Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent.*
- virtual unsigned char `getDataStructureType () const`
*Get the **DataStructure** (p. 1372) Type as defined in *CommandTypes.h*.*
- virtual std::string `toString () const`
*Returns a string containing the information for this **DataStructure** (p. 1372) such as its type and value of its elements.*
- virtual const std::string & `getPhysicalName () const`
Fetch this destination's physical name.
- virtual std::string & `getPhysicalName ()`
- virtual void `setPhysicalName (const std::string &physicalName)`
Set this destination's physical name.
- virtual bool `isAdvisory () const`
- virtual void `setAdvisory (bool advisory)`
- virtual bool `isConsumerAdvisory () const`
- virtual bool `isProducerAdvisory () const`
- virtual bool `isConnectionAdvisory () const`

- virtual bool **isExclusive** () const
- virtual void **setExclusive** (bool **exclusive**)
- virtual bool **isOrdered** () const
- virtual void **setOrdered** (bool **ordered**)
- virtual std::string **getOrderedTarget** () const
- virtual void **setOrderedTarget** (const std::string &**orderedTarget**)
- virtual **cms::Destination::DestinationType** **getDestinationType** () const =0
Returns the Type of Destination that this object represents.
- virtual bool **isTemporary** () const
Returns true if a temporary Destination.
- virtual bool **isTopic** () const
Returns true if a Topic Destination.
- virtual bool **isQueue** () const
Returns true if a Queue Destination.
- virtual bool **isComposite** () const
Returns true if this destination represents a collection of destinations; allowing a set of destinations to be published to or subscribed from in one CMS operation.
- virtual bool **isWildcard** () const
- const **activemq::util::ActiveMQProperties** & **getOptions** () const
- virtual const **cms::Destination** * **getCMSDestination** () const

Static Public Member Functions

- static std::string **createTemporaryName** (const std::string &clientId)
Create a temporary name from the clientId.
- static std::string **getClientId** (const **ActiveMQDestination** *destination)
From a temporary destination find the clientId of the Connection that created it.
- static **Pointer< ActiveMQDestination >** **createDestination** (int type, const std::string &name)
Creates a Destination given the String Name to use and a Type.

Static Public Attributes

- static const unsigned char **ID _ACTIVEMQDESTINATION** = 0

Protected Attributes

- bool **exclusive**
- bool **ordered**
- bool **advisory**
- std::string **orderedTarget**
- std::string **physicalName**
- **util::ActiveMQProperties** **options**

Static Protected Attributes

- static const std::string **ADVISORY_PREFIX**
prefix for Advisory message destinations
- static const std::string **CONSUMER_ADVISORY_PREFIX**
prefix for consumer advisory destinations
- static const std::string **PRODUCER_ADVISORY_PREFIX**
prefix for producer advisory destinations
- static const std::string **CONNECTION_ADVISORY_PREFIX**
prefix for connection advisory destinations
- static const std::string **DEFAULT_ORDERED_TARGET**
The default target for ordered destinations.
- static const std::string **TEMP_PREFIX**
- static const std::string **TEMP_POSTFIX**
- static const std::string **COMPOSITE_SEPARATOR**
- static const std::string **QUEUE_QUALIFIED_PREFIX**
- static const std::string **TOPIC_QUALIFIED_PREFIX**
- static const std::string **TEMP_QUEUE_QUALIFIED_PREFIX**
- static const std::string **TEMP_TOPIC_QUALIFIED_PREFIX**

6.28.1 Constructor & Destructor Documentation

6.28.1.1 `activemq::commands::ActiveMQDestination::ActiveMQDestination ()`

6.28.1.2 `activemq::commands::ActiveMQDestination::ActiveMQDestination (const std::string & physicalName)`

6.28.1.3 `virtual
activemq::commands::ActiveMQDestination::~~ActiveMQDestination ()
[inline, virtual]`

6.28.2 Member Function Documentation

6.28.2.1 `virtual ActiveMQDestination* ac-
tivemq::commands::ActiveMQDestination::cloneDataStructure () const
[inline, virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1373).

6.28.2.2 `virtual void activemq::commands::ActiveMQDestination::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Implements **activemq::commands::DataStructure** (p. 1374).

Referenced by **activemq::commands::ActiveMQTempDestination::copyDataStructure()**.

6.28.2.3 `static Pointer<ActiveMQDestination> activemq::commands::ActiveMQDestination::createDestination (int type, const std::string & name) [static]`

Creates a Destination given the String Name to use and a Type.

Parameters

type - The Type of Destination to Create

name - The Name to use in the creation of the Destination

Returns

Pointer to a new **ActiveMQDestination** (p. 240) instance.

6.28.2.4 `static std::string activemq::commands::ActiveMQDestination::createTemporaryName (const std::string & clientId) [inline, static]`

Create a temporary name from the clientId.

Parameters

clientId

Returns

6.28.2.5 `virtual bool activemq::commands::ActiveMQDestination::equals (const DataStructure * value) const [virtual]`

Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Implements **activemq::commands::DataStructure** (p. 1374).

Referenced by **activemq::commands::ActiveMQTopic::equals()**, and **activemq::commands::ActiveMQTempDestination::equals()**.

6.28.2.6 `static std::string activemq::commands::ActiveMQDestination::getClientId (const ActiveMQDestination * destination) [static]`

From a temporary destination find the clientId of the Connection that created it.

Parameters

destination

Returns

the clientId or null if not a temporary destination

6.28.2.7 `virtual const cms::Destination* activemq::commands::ActiveMQDestination::getCMSDestination () const [inline, virtual]`

Returns

the **cms::Destination** (p. 1387) interface pointer that the objects that derive from this class implement.

6.28.2.8 `virtual unsigned char activemq::commands::ActiveMQDestination::getDataStructureType () const [virtual]`

Get the **DataStructure** (p. 1372) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 1375).

6.28.2.9 `virtual cms::Destination::DestinationType activemq::commands::ActiveMQDestination::getDestinationType () const [pure virtual]`

Returns the Type of Destination that this object represents.

Returns

int type qualifier.

6.28.2.10 `const activemq::util::ActiveMQProperties& activemq::commands::ActiveMQDestination::getOptions () const`
[inline]

Returns

a reference (const) to the options properties for this Destination.

6.28.2.11 `virtual std::string activemq::commands::ActiveMQDestination::getOrderedTarget () const`
[inline, virtual]

Returns

Returns the orderedTarget.

6.28.2.12 `virtual const std::string& activemq::commands::ActiveMQDestination::getPhysicalName () const`
[inline, virtual]

Fetch this destination's physical name.

Returns

const string containing the name

6.28.2.13 `virtual std::string& activemq::commands::ActiveMQDestination::getPhysicalName ()`
[inline, virtual]

6.28.2.14 `virtual bool activemq::commands::ActiveMQDestination::isAdvisory () const` [inline, virtual]

Returns

Returns the advisory.

6.28.2.15 `virtual bool activemq::commands::ActiveMQDestination::isComposite () const` [inline, virtual]

Returns true if this destination represents a collection of destinations; allowing a set of destinations to be published to or subscribed from in one CMS operation.

Returns

true if this destination represents a collection of child destinations.

6.28.2.16 `virtual bool activemq::commands::ActiveMQDestination::isConnectionAdvisory ()`
`const [inline, virtual]`

Returns

true if this is a destination for Connection advisories

6.28.2.17 `virtual bool activemq::commands::ActiveMQDestination::isConsumerAdvisory ()`
`const [inline, virtual]`

Returns

true if this is a destination for Consumer advisories

6.28.2.18 `virtual bool activemq::commands::ActiveMQDestination::isExclusive ()`
`const [inline, virtual]`

Returns

Returns the exclusive.

6.28.2.19 `virtual bool activemq::commands::ActiveMQDestination::isOrdered ()`
`const [inline, virtual]`

Returns

Returns the ordered.

6.28.2.20 `virtual bool activemq::commands::ActiveMQDestination::isProducerAdvisory ()`
`const [inline, virtual]`

Returns

true if this is a destination for Producer advisories

6.28.2.21 `virtual bool activemq::commands::ActiveMQDestination::isQueue ()`
`const [inline, virtual]`

Returns true if a Queue Destination.

Returns

true/false

6.28.2.22 `virtual bool activemq::commands::ActiveMQDestination::isTemporary () const [inline, virtual]`

Returns true if a temporary Destination.

Returns

true/false

References cms::Destination::TEMPORARY_TOPIC.

6.28.2.23 `virtual bool activemq::commands::ActiveMQDestination::isTopic () const [inline, virtual]`

Returns true if a Topic Destination.

Returns

true/false

References cms::Destination::TOPIC.

6.28.2.24 `virtual bool activemq::commands::ActiveMQDestination::isWildcard () const [inline, virtual]`

Returns

true if the destination matches multiple possible destinations

6.28.2.25 `virtual void activemq::commands::ActiveMQDestination::setAdvisory (bool advisory) [inline, virtual]`

Parameters

advisory The advisory to set.

6.28.2.26 `virtual void activemq::commands::ActiveMQDestination::setExclusive (bool exclusive) [inline, virtual]`

Parameters

exclusive The exclusive to set.

6.28.2.27 `virtual void activemq::commands::ActiveMQDestination::setOrdered (bool ordered) [inline, virtual]`

Parameters

ordered The ordered to set.

6.28.2.28 `virtual void activemq::commands::ActiveMQDestination::setOrderedTarget (const std::string & orderedTarget)` [inline, virtual]

Parameters

orderedTarget The orderedTarget to set.

6.28.2.29 `virtual void activemq::commands::ActiveMQDestination::setPhysicalName (const std::string & physicalName)` [virtual]

Set this destination's physical name.

Returns

const string containing the name

6.28.2.30 `virtual std::string activemq::commands::ActiveMQDestination::toString () const` [virtual]

Returns a string containing the information for this **DataStructure** (p.1372) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from `activemq::commands::BaseDataStructure` (p.662).

6.28.3 Field Documentation

6.28.3.1 `bool activemq::commands::ActiveMQDestination::advisory` [protected]

6.28.3.2 `const std::string activemq::commands::ActiveMQDestination::ADVISORY_PREFIX` [static, protected]

prefix for Advisory message destinations

6.28.3.3 `const std::string activemq::commands::ActiveMQDestination::COMPOSITE_SEPARATOR` [static, protected]

6.28.3.4 `const std::string activemq::commands::ActiveMQDestination::CONNECTION_ADVISORY_PREFIX` [static, protected]

prefix for connection advisory destinations

6.28.3.5 `const std::string`
`activemq::commands::ActiveMQDestination::CONSUMER_`
`ADVISORY_PREFIX` [static, protected]

prefix for consumer advisory destinations

6.28.3.6 `const std::string`
`activemq::commands::ActiveMQDestination::DEFAULT_`
`ORDERED_TARGET` [static, protected]

The default target for ordered destinations.

6.28.3.7 `bool activemq::commands::ActiveMQDestination::exclusive` [protected]

6.28.3.8 `const unsigned char activemq::commands::ActiveMQDestination::ID_`
`ACTIVEMQDESTINATION = 0` [static]

6.28.3.9 `util::ActiveMQProperties ac-`
`tivemq::commands::ActiveMQDestination::options`
[protected]

6.28.3.10 `bool activemq::commands::ActiveMQDestination::ordered` [protected]

6.28.3.11 `std::string activemq::commands::ActiveMQDestination::orderedTarget`
[protected]

6.28.3.12 `std::string activemq::commands::ActiveMQDestination::physicalName`
[protected]

6.28.3.13 `const std::string`
`activemq::commands::ActiveMQDestination::PRODUCER_`
`ADVISORY_PREFIX` [static, protected]

prefix for producer advisory destinations

- 6.28.3.14 `const std::string activemq::commands::ActiveMQDestination::QUEUE_ - QUALIFIED_PREFIX` [static, protected]
- 6.28.3.15 `const std::string activemq::commands::ActiveMQDestination::TEMP_ - POSTFIX` [static, protected]
- 6.28.3.16 `const std::string activemq::commands::ActiveMQDestination::TEMP_ - PREFIX` [static, protected]
- 6.28.3.17 `const std::string activemq::commands::ActiveMQDestination::TEMP_ - QUEUE_QUALIFIED_PREFIX` [static, protected]
- 6.28.3.18 `const std::string activemq::commands::ActiveMQDestination::TEMP_ - TOPIC_QUALIFIED_PREFIX` [static, protected]
- 6.28.3.19 `const std::string activemq::commands::ActiveMQDestination::TOPIC_ - QUALIFIED_PREFIX` [static, protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ActiveMQDestination.h`

6.29 `activemq::wireformat::openwire::marshal::v3::ActiveMQDestination` Class Reference

Marshaling code for Open Wire Format for `ActiveMQDestinationMarshaller` (p. 250).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQDestinationMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller`:

Public Member Functions

- `ActiveMQDestinationMarshaller ()`
- `virtual ~ActiveMQDestinationMarshaller ()`
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)`

Write a object instance to data output stream.

- virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.29.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 250).
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.29.2 Constructor & Destructor Documentation

6.29.2.1 activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller::ActiveMQDestinationMarshaller () [inline]

6.29.2.2 virtual activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller::~~ActiveMQDestinationMarshaller () [inline, virtual]

6.29.3 Member Function Documentation

6.29.3.1 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1342).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller** (p. 384), **activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller**

(p. 460), `activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller` (p. 483), `activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller` (p. 507), and `activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller` (p. 555).

6.29.3.2 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1348).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller` (p. 385), `activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller` (p. 460), `activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller` (p. 483), `activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller` (p. 507), and `activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller` (p. 555).

6.29.3.3 `virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1354).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller** (p. 385), **activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller** (p. 461), **activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller** (p. 483), **activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller** (p. 507), and **activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller** (p. 556).

6.29.3.4 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller::tightMarshal2
(OpenWireFormat * *wireFormat*, commands::DataStructure
*** *dataStructure*, decaf::io::DataOutputStream * *dataOut*,**
utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1360).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller** (p. 385), **activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller** (p. 461), **activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller** (p. 484), **activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller** (p. 508), and **activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller** (p. 556).

6.29.3.5 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller::tightUnmarshal
(OpenWireFormat * *wireFormat*, commands::DataStructure
*** *dataStructure*, decaf::io::DataInputStream * *dataIn*,**
utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1366).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller` (p. 386), `activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller` (p. 462), `activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller` (p. 484), `activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller` (p. 508), and `activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller` (p. 557).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQDestinationMarshaller.h`

6.30 `activemq::wireformat::openwire::marshal::v1::ActiveMQDestination` Class Reference

Marshaling code for Open Wire Format for `ActiveMQDestinationMarshaller` (p. 254).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQDestinationMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller`:

Public Member Functions

- `ActiveMQDestinationMarshaller ()`
- `virtual ~ActiveMQDestinationMarshaller ()`
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`

Un-marshal an object instance from the data input stream.

- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`

Write the booleans that this object uses to a BooleanStream.

- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)`

Write a object instance to data output stream.

- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)`

Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.30.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 254).
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.30.2 Constructor & Destructor Documentation

6.30.2.1 activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller::ActiveMQDestinationMarshaller () [inline]

6.30.2.2 virtual activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller::~~ActiveMQDestinationMarshaller () [inline, virtual]

6.30.3 Member Function Documentation

6.30.3.1 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1342).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller** (p. 388), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller** (p. 463), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller** (p. 487), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller** (p. 511), and **activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller** (p. 559).

```

6.30.3.2 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller::looseUnmarsh
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]

```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1348).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller` (p. 389), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller` (p. 464), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller` (p. 487), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller` (p. 511), and `activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller` (p. 559).

```

6.30.3.3 virtual int ac-
tivemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller::tightMarshall
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException
) [virtual]

```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1354).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller` (p. 389), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller` (p. 464), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller` (p. 487), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller` (p. 511), and `activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller` (p. 560).

6.30.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1360).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller` (p. 389), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller` (p. 465), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller` (p. 488), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller` (p. 512), and `activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller` (p. 560).

6.30.3.5 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1366).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller` (p. 390), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller` (p. 465), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller` (p. 488), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller`

(p. 512), and `activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller` (p. 560).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQDestinationMarshaller.h`

6.31 `activemq::wireformat::openwire::marshal::v4::ActiveMQDestination` Class Reference

Marshaling code for Open Wire Format for `ActiveMQDestinationMarshaller` (p. 258).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQDestinationMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller`:

Public Member Functions

- `ActiveMQDestinationMarshaller ()`
- `virtual ~ActiveMQDestinationMarshaller ()`
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`

Un-marshal an object instance from the data input stream.

- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`

Write the booleans that this object uses to a BooleanStream.

- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)`

Write a object instance to data output stream.

- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)`

Un-marshal an object instance from the data input stream.

- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)`

Write a object instance to data output stream.

6.31.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 258).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.31.2 Constructor & Destructor Documentation

6.31.2.1 `activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller::ActiveMQDestinationMarshaller()` [inline]

6.31.2.2 `virtual activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller::~~ActiveMQDestinationMarshaller()` [inline, virtual]

6.31.3 Member Function Documentation

6.31.3.1 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut)` throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryWriter that provides that data sink

Exceptions

- IOException* if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1342).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller` (p. 392), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller` (p. 467), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller` (p. 491), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller` (p. 515), and `activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller` (p. 563).

6.31.3.2 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn)` throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1348).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller` (p. 393), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller` (p. 468), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller` (p. 491), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller` (p. 515), and `activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller` (p. 563).

```
6.31.3.3 virtual int ac-
tivemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException
) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1354).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller` (p. 393), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller` (p. 468), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller` (p. 491), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller` (p. 515), and `activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller` (p. 563).

```
6.31.3.4 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions

- IOException* if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1360).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller` (p. 393), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller` (p. 469), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller` (p. 492), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller` (p. 516), and `activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller` (p. 564).

6.31.3.5 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1366).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller` (p. 394), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller` (p. 469), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller` (p. 492), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller` (p. 516), and `activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller` (p. 564).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQDestinationMarshaller.h`

6.32 activemq::wireformat::openwire::marshal::v5::ActiveMQDestination Class Reference

Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 262).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQDestinationMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller`:

Public Member Functions

- **ActiveMQDestinationMarshaller** ()
- virtual **~ActiveMQDestinationMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

6.32.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 262).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module

6.32.2 Constructor & Destructor Documentation

6.32.2.1 `activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller::ActiveMQDestinationMarshaller () [inline]`

6.32.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller::~~ActiveMQDestinationMarshaller () [inline, virtual]`

6.32.3 Member Function Documentation

6.32.3.1 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1342).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller` (p. 396), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller` (p. 471), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller` (p. 495), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller` (p. 519), and `activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller` (p. 567).

6.32.3.2 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataIn* - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1348).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller` (p. 397), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller` (p. 471), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller` (p. 495), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller` (p. 519), and `activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller` (p. 567).

```
6.32.3.3 virtual int ac-
          tivemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller::tightMarshal1
          ( OpenWireFormat * wireFormat, commands::DataStructure *
            dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException
          ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1354).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller` (p. 397), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller` (p. 472), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller` (p. 495), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller` (p. 519), and `activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller` (p. 567).

```
6.32.3.4 virtual void ac-
          tivemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller::tightMarshal2
          ( OpenWireFormat * wireFormat, commands::DataStructure
            * dataStructure, decaf::io::DataOutputStream * dataOut,
            utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1360).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller** (p. 397), **activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller** (p. 472), **activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller** (p. 496), **activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller** (p. 520), and **activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller** (p. 568).

6.32.3.5 virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1366).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller** (p. 398), **activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller** (p. 473), **activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller** (p. 496), **activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller** (p. 520), and **activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller** (p. 568).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQDestinationMarshaller.h`

6.33 activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 265).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQDestinationMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller`:

Public Member Functions

- **ActiveMQDestinationMarshaller** ()
- virtual **~ActiveMQDestinationMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

6.33.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 265).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module

6.33.2 Constructor & Destructor Documentation

6.33.2.1 `activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller::ActiveMQDestinationMarshaller () [inline]`

6.33.2.2 `virtual activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller::~~ActiveMQDestinationMarshaller () [inline, virtual]`

6.33.3 Member Function Documentation

6.33.3.1 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller::marshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1342).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller` (p. 400), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller` (p. 474), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller` (p. 499), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller` (p. 523), and `activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller` (p. 571).

6.33.3.2 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller::unmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataIn* - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1348).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller` (p. 401), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller` (p. 475), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller` (p. 499), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller` (p. 523), and `activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller` (p. 571).

```
6.33.3.3 virtual int ac-
tivemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller::tightMarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException
) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1354).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller` (p. 401), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller` (p. 475), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller` (p. 499), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller` (p. 523), and `activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller` (p. 571).

```
6.33.3.4 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1360).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller` (p. 401), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller` (p. 476), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller` (p. 500), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller` (p. 524), and `activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller` (p. 572).

```
6.33.3.5 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller::tightUnmarsh
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1366).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller` (p. 402), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller` (p. 476), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller` (p. 500), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller` (p. 524), and `activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller` (p. 572).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQDestinationMarshaller.h`

6.34 activemq::exceptions::ActiveMQException Class Reference

```
#include <src/main/activemq/exceptions/ActiveMQException.h>
```

Inheritance diagram for `activemq::exceptions::ActiveMQException`:

Public Member Functions

- **ActiveMQException** () throw ()
Default Constructor.
- **ActiveMQException** (const **ActiveMQException** &ex) throw ()
Copy Constructor.
- **ActiveMQException** (const **decaf::lang::Exception** &ex) throw ()
Copy Constructor.
- **ActiveMQException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual ~**ActiveMQException** () throw ()
- virtual **ActiveMQException** * **clone** () const
Clones this exception.
- virtual **cms::CMSEException** **convertToCMSEException** () const
Converts this exception to a new CMSEException.

6.34.1 Constructor & Destructor Documentation

6.34.1.1 **activemq::exceptions::ActiveMQException::ActiveMQException** () throw ()

Default Constructor.

6.34.1.2 **activemq::exceptions::ActiveMQException::ActiveMQException** (const **ActiveMQException** & *ex*) throw ()

Copy Constructor.

Parameters

ex The Exception whose internal data is copied into this instance.

6.34.1.3 **activemq::exceptions::ActiveMQException::ActiveMQException** (const **decaf::lang::Exception** & *ex*) throw ()

Copy Constructor.

Parameters

ex The Exception whose internal data is copied into this instance.

6.34.1.4 `activemq::exceptions::ActiveMQException::ActiveMQException (const char * file, const int lineNumber, const char * msg, ...) throw ()`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message.

Parameters

file The file name where exception occurs.

lineNumber The line number where the exception occurred.

msg The message to report.

... The list of primitives that are formatted into the message.

6.34.1.5 `virtual activemq::exceptions::ActiveMQException::~~ActiveMQException () throw () [virtual]`

6.34.2 Member Function Documentation

6.34.2.1 `virtual ActiveMQException* activemq::exceptions::ActiveMQException::clone () const [virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

Copy of this Exception object

Reimplemented from `decaf::lang::Exception` (p. 1479).

Reimplemented in `activemq::exceptions::BrokerException` (p. 691).

6.34.2.2 `virtual cms::CMSException activemq::exceptions::ActiveMQException::convertToCMSException () const [virtual]`

Converts this exception to a new CMSException.

Returns

a CMSException with the data from this exception

The documentation for this class was generated from the following file:

- `src/main/activemq/exceptions/ActiveMQException.h`

6.35 activemq::commands::ActiveMQMapMessage Class Reference

```
#include <src/main/activemq/commands/ActiveMQMapMessage.h>
```

Inheritance diagram for activemq::commands::ActiveMQMapMessage:

Public Member Functions

- **ActiveMQMapMessage** ()
- virtual **~ActiveMQMapMessage** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual bool **isMarshalAware** () const
Determine if this object is aware of marshaling and should have its before and after marshaling methods called.
- virtual **ActiveMQMapMessage** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual void **beforeMarshal** (**wireformat::WireFormat** *wireFormat) throw (decaf::io::IOException)
Perform any processing needed before an marshal.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1372) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent.*
- virtual void **clearBody** () throw (cms::CMSEException)
Clears out the body of the message.
- virtual **cms::MapMessage** * **clone** () const
Clone this message exactly, returns a new instance that the caller is required to delete.
- virtual std::vector< std::string > **getMapNames** () const throw (cms::CMSEException)
Returns an Enumeration of all the names in the MapMessage object.
- virtual bool **itemExists** (const std::string &name) const throw (cms::CMSEException)
Indicates whether an item exists in this MapMessage object.

- virtual bool **getBoolean** (const std::string &name) const throw (cms::MessageFormatException, cms::CMSEException)
Returns the Boolean value of the Specified name.
- virtual void **setBoolean** (const std::string &name, bool value) throw (cms::MessageNotWriteableException, cms::CMSEException)
Sets a boolean value with the specified name into the Map.
- virtual unsigned char **getByte** (const std::string &name) const throw (cms::MessageFormatException, cms::CMSEException)
Returns the Byte value of the Specified name.
- virtual void **setByte** (const std::string &name, unsigned char value) throw (cms::MessageNotWriteableException, cms::CMSEException)
Sets a Byte value with the specified name into the Map.
- virtual std::vector< unsigned char > **getBytes** (const std::string &name) const throw (cms::MessageFormatException, cms::CMSEException)
Returns the Bytes value of the Specified name.
- virtual void **setBytes** (const std::string &name, const std::vector< unsigned char > &value) throw (cms::MessageNotWriteableException, cms::CMSEException)
Sets a Bytes value with the specified name into the Map.
- virtual char **getChar** (const std::string &name) const throw (cms::MessageFormatException, cms::CMSEException)
Returns the Char value of the Specified name.
- virtual void **setChar** (const std::string &name, char value) throw (cms::MessageNotWriteableException, cms::CMSEException)
Sets a Char value with the specified name into the Map.
- virtual double **getDouble** (const std::string &name) const throw (cms::MessageFormatException, cms::CMSEException)
Returns the Double value of the Specified name.
- virtual void **setDouble** (const std::string &name, double value) throw (cms::MessageNotWriteableException, cms::CMSEException)
Sets a Double value with the specified name into the Map.
- virtual float **getFloat** (const std::string &name) const throw (cms::MessageFormatException, cms::CMSEException)
Returns the Float value of the Specified name.
- virtual void **setFloat** (const std::string &name, float value) throw (cms::MessageNotWriteableException, cms::CMSEException)
Sets a Float value with the specified name into the Map.
- virtual int **getInt** (const std::string &name) const throw (cms::MessageFormatException, cms::CMSEException)
Returns the Int value of the Specified name.

- virtual void **setInt** (const std::string &name, int value) throw (cms::MessageNotWriteableException, cms::CMSEException)
Sets a Int value with the specified name into the Map.
- virtual long long **getLong** (const std::string &name) const throw (cms::MessageFormatException, cms::CMSEException)
Returns the Long value of the Specified name.
- virtual void **setLong** (const std::string &name, long long value) throw (cms::MessageNotWriteableException, cms::CMSEException)
Sets a Long value with the specified name into the Map.
- virtual short **getShort** (const std::string &name) const throw (cms::MessageFormatException, cms::CMSEException)
Returns the Short value of the Specified name.
- virtual void **setShort** (const std::string &name, short value) throw (cms::MessageNotWriteableException, cms::CMSEException)
Sets a Short value with the specified name into the Map.
- virtual std::string **getString** (const std::string &name) const throw (cms::MessageFormatException, cms::CMSEException)
Returns the String value of the Specified name.
- virtual void **setString** (const std::string &name, const std::string &value) throw (cms::MessageNotWriteableException, cms::CMSEException)
Sets a String value with the specified name into the Map.

Static Public Attributes

- static const unsigned char **ID_ACTIVEMQMAPMESSAGE** = 25

Protected Member Functions

- **util::PrimitiveMap & getMap** () throw (decaf::lang::exceptions::NullPointerException)
Fetches a reference to this objects PrimitiveMap, if one needs to be created or unmarshaled, this will perform the correct steps.
- const **util::PrimitiveMap & getMap** () const throw (decaf::lang::exceptions::NullPointerException)
- virtual void **checkMapIsUnmarshalled** () const throw (decaf::lang::exceptions::NullPointerException)
Performs the unmarshal on the Map if needed, otherwise just returns.

6.35.1 Constructor & Destructor Documentation

6.35.1.1 `activemq::commands::ActiveMQMapMessage::ActiveMQMapMessage ()`

6.35.1.2 `virtual
activemq::commands::ActiveMQMapMessage::~~ActiveMQMapMessage ()` [virtual]

6.35.2 Member Function Documentation

6.35.2.1 `virtual void activemq::commands::ActiveMQMapMessage::beforeMarshal (wireformat::WireFormat * wireFormat) throw (decaf::io::IOException)` [virtual]

Perform any processing needed before an marshal.

Parameters

wireFormat - the OpenWireFormat object in use.

Implements `activemq::wireformat::MarshalAware` (p. 1994).

6.35.2.2 `virtual void activemq::commands::ActiveMQMapMessage::checkMapIsUnmarshalled () const throw (decaf::lang::exceptions::NullPointerException)` [protected, virtual]

Performs the unmarshal on the Map if needed, otherwise just returns.

6.35.2.3 `virtual void activemq::commands::ActiveMQMapMessage::clearBody () throw (cms::CMSException)` [virtual]

Clears out the body of the message.

This does not clear the headers or properties.

Reimplemented from `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 331).

6.35.2.4 `virtual cms::MapMessage* activemq::commands::ActiveMQMapMessage::clone () const` [inline, virtual]

Clone this message exactly, returns a new instance that the caller is required to delete.

Returns

new copy of this message

6.35.2.5 `virtual ActiveMQMapMessage* activemq::commands::ActiveMQMapMessage::cloneDataStructure ()`
`const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Reimplemented from `activemq::commands::Message` (p. 2023).

6.35.2.6 `virtual void activemq::commands::ActiveMQMapMessage::copyDataStructure (const DataStructure * src)`
`[virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Reimplemented from `activemq::commands::Message` (p. 2023).

6.35.2.7 `virtual bool activemq::commands::ActiveMQMapMessage::equals (const DataStructure * value)`
`const [virtual]`

Compares the `DataStructure` (p. 1372) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if `DataStructure`'s are Equal.

Reimplemented from `activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 331).

6.35.2.8 `virtual bool activemq::commands::ActiveMQMapMessage::getBoolean (const std::string & name)`
`const throw (cms::MessageFormatException, cms::CMSException) [virtual]`

Returns the Boolean value of the Specified name.

Parameters

name Name of the value to fetch from the map

Exceptions

CMSException - if the operation fails due to an internal error.

MessageFormatException - if this type conversion is invalid.

6.35.2.9 `virtual unsigned char activemq::commands::ActiveMQMapMessage::getByte (const std::string & name) const throw (cms::MessageFormatException, cms::CMSException) [virtual]`

Returns the Byte value of the Specified name.

Parameters

name Name of the value to fetch from the map

Exceptions

CMSException - if the operation fails due to an internal error.

MessageFormatException - if this type conversion is invalid.

6.35.2.10 `virtual std::vector<unsigned char> activemq::commands::ActiveMQMapMessage::getBytes (const std::string & name) const throw (cms::MessageFormatException, cms::CMSException) [virtual]`

Returns the Bytes value of the Specified name.

Parameters

name Name of the value to fetch from the map

Exceptions

CMSException - if the operation fails due to an internal error.

MessageFormatException - if this type conversion is invalid.

6.35.2.11 `virtual char activemq::commands::ActiveMQMapMessage::getChar (const std::string & name) const throw (cms::MessageFormatException, cms::CMSException) [virtual]`

Returns the Char value of the Specified name.

Parameters

name name of the value to fetch from the map

Exceptions

CMSException - if the operation fails due to an internal error.

MessageFormatException - if this type conversion is invalid.

6.35.2.12 `virtual unsigned char activemq::commands::ActiveMQMapMessage::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataStructure** (p. 1372) type copy.

Reimplemented from **activemq::commands::Message** (p. 2025).

6.35.2.13 `virtual double activemq::commands::ActiveMQMapMessage::getDouble (const std::string & name) const throw (cms::MessageFormatException, cms::CMSException) [virtual]`

Returns the Double value of the Specified name.

Parameters

name Name of the value to fetch from the map

Exceptions

CMSException - if the operation fails due to an internal error.

MessageFormatException - if this type conversion is invalid.

6.35.2.14 `virtual float activemq::commands::ActiveMQMapMessage::getFloat (const std::string & name) const throw (cms::MessageFormatException, cms::CMSException) [virtual]`

Returns the Float value of the Specified name.

Parameters

name Name of the value to fetch from the map

Exceptions

CMSException - if the operation fails due to an internal error.

MessageFormatException - if this type conversion is invalid.

6.35.2.15 `virtual int activemq::commands::ActiveMQMapMessage::getInt (const std::string & name) const throw (cms::MessageFormatException, cms::CMSException) [virtual]`

Returns the Int value of the Specified name.

Parameters

name Name of the value to fetch from the map

Exceptions

CMSEException - if the operation fails due to an internal error.

MessageFormatException - if this type conversion is invalid.

6.35.2.16 `virtual long long activemq::commands::ActiveMQMapMessage::getLong (const std::string & name) const throw (cms::MessageFormatException, cms::CMSEException) [virtual]`

Returns the Long value of the Specified name.

Parameters

name Name of the value to fetch from the map

Exceptions

CMSEException - if the operation fails due to an internal error.

MessageFormatException - if this type conversion is invalid.

6.35.2.17 `util::PrimitiveMap& activemq::commands::ActiveMQMapMessage::getMap () throw (decaf::lang::exceptions::NullPointerException) [protected]`

Fetches a reference to this objects PrimitiveMap, if one needs to be created or unmarshaled, this will perform the correct steps.

Returns

reference to a PrimitiveMap.

6.35.2.18 `const util::PrimitiveMap& activemq::commands::ActiveMQMapMessage::getMap () const throw (decaf::lang::exceptions::NullPointerException) [protected]`

6.35.2.19 `virtual std::vector< std::string > activemq::commands::ActiveMQMapMessage::getMapNames () const throw (cms::CMSEException) [virtual]`

Returns an Enumeration of all the names in the MapMessage object.

Returns

STL Vector of String values, each of which is the name of an item in the MapMessage

Exceptions

CMSEException - if the operation fails due to an internal error.

6.35.2.20 `virtual short activemq::commands::ActiveMQMapMessage::getShort (const std::string & name) const throw (cms::MessageFormatException, cms::CMSException) [virtual]`

Returns the Short value of the Specified name.

Parameters

name Name of the value to fetch from the map

Exceptions

CMSException - if the operation fails due to an internal error.

MessageFormatException - if this type conversion is invalid.

6.35.2.21 `virtual std::string activemq::commands::ActiveMQMapMessage::getString (const std::string & name) const throw (cms::MessageFormatException, cms::CMSException) [virtual]`

Returns the String value of the Specified name.

Parameters

name Name of the value to fetch from the map

Exceptions

CMSException - if the operation fails due to an internal error.

MessageFormatException - if this type conversion is invalid.

6.35.2.22 `virtual bool activemq::commands::ActiveMQMapMessage::isMarshalAware () const [inline, virtual]`

Determine if this object is aware of marshaling and should have its before and after marshaling methods called.

Defaults to false.

Returns

true if aware of marshaling

Reimplemented from `activemq::commands::Message` (p. 2029).

6.35.2.23 `virtual bool activemq::commands::ActiveMQMapMessage::itemExists (const std::string & name) const throw (cms::CMSException) [virtual]`

Indicates whether an item exists in this MapMessage object.

Parameters

name String name of the Object in question

Returns

boolean value indicating if the name is in the map

Exceptions

CMSEException - if the operation fails due to an internal error.

6.35.2.24 virtual void activemq::commands::ActiveMQMapMessage::setBoolean
(const std::string & *name*, bool *value*) throw (cms::MessageNotWriteableException, cms::CMSEException) [virtual]

Sets a boolean value with the specified name into the Map.

Parameters

name the name of the boolean

value the boolean value to set in the Map

Exceptions

CMSEException - if the operation fails due to an internal error.

MessageNotWriteableException - if the Message (p. 2018) is in Read-only Mode.

6.35.2.25 virtual void activemq::commands::ActiveMQMapMessage::setByte
(const std::string & *name*, unsigned char *value*) throw (cms::MessageNotWriteableException, cms::CMSEException) [virtual]

Sets a Byte value with the specified name into the Map.

Parameters

name the name of the Byte

value the Byte value to set in the Map

Exceptions

CMSEException - if the operation fails due to an internal error.

MessageNotWriteableException - if the Message (p. 2018) is in Read-only Mode.

6.35.2.26 virtual void activemq::commands::ActiveMQMapMessage::setBytes (const std::string & *name*, const std::vector< unsigned char > & *value*) throw (cms::MessageNotWriteableException, cms::CMSEException) [virtual]

Sets a Bytes value with the specified name into the Map.

Parameters

name The name of the Bytes
value The Bytes value to set in the Map

Exceptions

CMSEException - if the operation fails due to an internal error.
MessageNotWriteableException - if the **Message** (p. 2018) is in Read-only Mode.

6.35.2.27 `virtual void activemq::commands::ActiveMQMapMessage::setChar
 (const std::string & name, char value) throw (
 cms::MessageNotWriteableException, cms::CMSEException) [virtual]`

Sets a Char value with the specified name into the Map.

Parameters

name the name of the Char
value the Char value to set in the Map

Exceptions

CMSEException - if the operation fails due to an internal error.
MessageNotWriteableException - if the **Message** (p. 2018) is in Read-only Mode.

6.35.2.28 `virtual void activemq::commands::ActiveMQMapMessage::setDouble
 (const std::string & name, double value) throw (
 cms::MessageNotWriteableException, cms::CMSEException) [virtual]`

Sets a Double value with the specified name into the Map.

Parameters

name The name of the Double
value The Double value to set in the Map

Exceptions

CMSEException - if the operation fails due to an internal error.
MessageNotWriteableException - if the **Message** (p. 2018) is in Read-only Mode.

6.35.2.29 `virtual void activemq::commands::ActiveMQMapMessage::setFloat
 (const std::string & name, float value) throw (
 cms::MessageNotWriteableException, cms::CMSEException) [virtual]`

Sets a Float value with the specified name into the Map.

Parameters

name The name of the Float

value The Float value to set in the Map

Exceptions

CMSEException - if the operation fails due to an internal error.

MessageNotWriteableException - if the `Message` (p. 2018) is in Read-only Mode.

6.35.2.30 `virtual void activemq::commands::ActiveMQMapMessage::setInt (const std::string & name, int value) throw (cms::MessageNotWriteableException, cms::CMSEException) [virtual]`

Sets a Int value with the specified name into the Map.

Parameters

name The name of the Int

value The Int value to set in the Map

Exceptions

CMSEException - if the operation fails due to an internal error.

MessageNotWriteableException - if the `Message` (p. 2018) is in Read-only Mode.

6.35.2.31 `virtual void activemq::commands::ActiveMQMapMessage::setLong (const std::string & name, long long value) throw (cms::MessageNotWriteableException, cms::CMSEException) [virtual]`

Sets a Long value with the specified name into the Map.

Parameters

name The name of the Long

value The Long value to set in the Map

Exceptions

CMSEException - if the operation fails due to an internal error.

MessageNotWriteableException - if the `Message` (p. 2018) is in Read-only Mode.

6.35.2.32 `virtual void activemq::commands::ActiveMQMapMessage::setShort (const std::string & name, short value) throw (cms::MessageNotWriteableException, cms::CMSEException) [virtual]`

Sets a Short value with the specified name into the Map.

Parameters

name The name of the Short

value The Short value to set in the Map

Exceptions

CMSEException - if the operation fails due to an internal error.

MessageNotWriteableException - if the `Message` (p. 2018) is in Read-only Mode.

6.35.2.33 `virtual void activemq::commands::ActiveMQMapMessage::setString
(const std::string & name, const std::string & value) throw (cms::MessageNotWriteableException, cms::CMSException)` [virtual]

Sets a String value with the specified name into the Map.

Parameters

name The name of the String

value The String value to set in the Map

Exceptions

CMSException - if the operation fails due to an internal error.

MessageNotWriteableException - if the `Message` (p. 2018) is in Read-only Mode.

6.35.2.34 `virtual std::string activemq::commands::ActiveMQMapMessage::toString
() const` [virtual]

Returns a string containing the information for this `DataStructure` (p. 1372) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from `activemq::commands::Message` (p. 2033).

6.35.3 Field Documentation

6.35.3.1 `const unsigned char activemq::commands::ActiveMQMapMessage::ID_-
ACTIVEMQMAPMESSAGE = 25` [static]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ActiveMQMapMessage.h`

6.36 activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessage Class Reference

Marshaling code for Open Wire Format for `ActiveMQMapMessageMarshaller` (p. 284).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQMapMessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller`:

Public Member Functions

- **ActiveMQMapMessageMarshaller** ()
- virtual **~ActiveMQMapMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.36.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p.284).
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.36.2 Constructor & Destructor Documentation

6.36.2.1 `activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller::ActiveMQMapMessageMarshaller () [inline]`

6.36.2.2 `virtual activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller::~~ActiveMQMapMessageMarshaller () [inline, virtual]`

6.36.3 Member Function Documentation

6.36.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.36.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.36.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2176).

6.36.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2176).

6.36.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2177).

6.36.3.6 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2178).

```
6.36.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller::tightUnmars
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2178).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQMapMessageMarshaller.h`

6.37 `activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller` Class Reference

Marshaling code for Open Wire Format for `ActiveMQMapMessageMarshaller` (p. 288).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQMapMessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller`:

Public Member Functions

- **ActiveMQMapMessageMarshaller** ()
- virtual **~ActiveMQMapMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.37.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p.288).
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.37.2 Constructor & Destructor Documentation

6.37.2.1 `activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller::ActiveMQMapMessageMarshaller () [inline]`

6.37.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller::~~ActiveMQMapMessageMarshaller () [inline, virtual]`

6.37.3 Member Function Documentation

6.37.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.37.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.37.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2184).

6.37.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2184).

6.37.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2185).

6.37.3.6 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2186).

```
6.37.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller::tightUnmars
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2186).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQMapMessageMarshaller.h`

6.38 `activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller` Class Reference

Marshaling code for Open Wire Format for `ActiveMQMapMessageMarshaller` (p. 292).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQMapMessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller`:

Public Member Functions

- **ActiveMQMapMessageMarshaller** ()
- virtual **~ActiveMQMapMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.38.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p.292).
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.38.2 Constructor & Destructor Documentation

6.38.2.1 `activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller::ActiveMQMapMessageMarshaller () [inline]`

6.38.2.2 `virtual activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller::~~ActiveMQMapMessageMarshaller () [inline, virtual]`

6.38.3 Member Function Documentation

6.38.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.38.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.38.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2172).

6.38.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2172).

6.38.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2173).

6.38.3.6 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2174).

```
6.38.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller::tightUnmars
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2174).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQMapMessageMarshaller.h`

6.39 `activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller` Class Reference

Marshaling code for Open Wire Format for `ActiveMQMapMessageMarshaller` (p. 296).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQMapMessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller`:

Public Member Functions

- **ActiveMQMapMessageMarshaller** ()
- virtual **~ActiveMQMapMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.39.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p.296).
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.39.2 Constructor & Destructor Documentation

6.39.2.1 `activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller::ActiveMQMapMessageMarshaller () [inline]`

6.39.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller::~~ActiveMQMapMessageMarshaller () [inline, virtual]`

6.39.3 Member Function Documentation

6.39.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.39.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.39.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2180).

6.39.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2180).

6.39.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2181).

6.39.3.6 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2182).

```
6.39.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller::tightUnmars
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2182).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQMapMessageMarshaller.h`

6.40 `activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller` Class Reference

Marshaling code for Open Wire Format for `ActiveMQMapMessageMarshaller` (p. 300).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQMapMessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller`:

Public Member Functions

- **ActiveMQMapMessageMarshaller** ()
- virtual **~ActiveMQMapMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.40.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p.300).
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.40.2 Constructor & Destructor Documentation

6.40.2.1 `activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller::ActiveMQMapMessageMarshaller()` [inline]

6.40.2.2 `virtual activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller::~~ActiveMQMapMessageMarshaller()` [inline, virtual]

6.40.3 Member Function Documentation

6.40.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.40.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.40.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut)` throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2167).

6.40.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2168).

6.40.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2169).

6.40.3.6 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions

- IOException* if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2169).

```
6.40.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller::tightUnmars
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2170).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQMapMessageMarshaller.h`

6.41 `activemq::commands::ActiveMQMessage` Class Reference

```
#include <src/main/activemq/commands/ActiveMQMessage.h>
```

Inheritance diagram for `activemq::commands::ActiveMQMessage`:

Public Member Functions

- **ActiveMQMessage** ()
- virtual **~ActiveMQMessage** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual **ActiveMQMessage** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1372) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent.*
- virtual **cms::Message** * **clone** () const
Clone this message exactly, returns a new instance that the caller is required to delete.

Static Public Attributes

- static const unsigned char **ID_ACTIVEMQMESSAGE** = 23

6.41.1 Constructor & Destructor Documentation

- 6.41.1.1 **activemq::commands::ActiveMQMessage::ActiveMQMessage** ()
- 6.41.1.2 **virtual activemq::commands::ActiveMQMessage::~~ActiveMQMessage** () [inline, virtual]

6.41.2 Member Function Documentation

- 6.41.2.1 **virtual cms::Message* activemq::commands::ActiveMQMessage::clone** () const [inline, virtual]

Clone this message exactly, returns a new instance that the caller is required to delete.

Returns

new copy of this message

6.41.2.2 `virtual ActiveMQMessage* activemq::commands::ActiveMQMessage::cloneDataStructure () const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Reimplemented from `activemq::commands::Message` (p. 2023).

6.41.2.3 `virtual void activemq::commands::ActiveMQMessage::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Reimplemented from `activemq::commands::Message` (p. 2023).

6.41.2.4 `virtual bool activemq::commands::ActiveMQMessage::equals (const DataStructure * value) const [virtual]`

Compares the `DataStructure` (p. 1372) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 331).

6.41.2.5 `virtual unsigned char activemq::commands::ActiveMQMessage::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

Returns

new `DataStructure` (p. 1372) type copy.

Reimplemented from `activemq::commands::Message` (p. 2025).

6.41.2.6 `virtual std::string activemq::commands::ActiveMQMessage::toString ()`
`const [virtual]`

Returns a string containing the information for this **DataSet** (p.1372) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from `activemq::commands::Message` (p. 2033).

6.41.3 Field Documentation

6.41.3.1 `const unsigned char activemq::commands::ActiveMQMessage::ID_ -`
`ACTIVEMQMESSAGE = 23 [static]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ActiveMQMessage.h`

6.42 `activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller` Class Reference

Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 307).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQMessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller`:

Public Member Functions

- **ActiveMQMessageMarshaller** ()
- `virtual ~ActiveMQMessageMarshaller` ()
- `virtual commands::DataSet * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaler.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataSet *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataSet *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.42.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 307). NOTE! This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.42.2 Constructor & Destructor Documentation

6.42.2.1 `activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller::ActiveMQMessageMarshaller () [inline]`

6.42.2.2 `virtual activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller::~~ActiveMQMessageMarshaller () [inline, virtual]`

6.42.3 Member Function Documentation

6.42.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.42.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.42.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2176).

6.42.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2176).

6.42.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2177).

```
6.42.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2178).

```
6.42.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2178).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQMessageMarshaller.h`

6.43 `activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMa` Class Reference

Marshaling code for Open Wire Format for `ActiveMQMessageMarshaller` (p. 311).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQMessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller`:

Public Member Functions

- `ActiveMQMessageMarshaller ()`
- `virtual ~ActiveMQMessageMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaler.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write a object instance to data output stream.

- virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.43.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p.311). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.43.2 Constructor & Destructor Documentation

6.43.2.1 `activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller::ActiveMQMessageMarshaller () [inline]`

6.43.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller::~~ActiveMQMessageMarshaller () [inline, virtual]`

6.43.3 Member Function Documentation

6.43.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p.1331).

6.43.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p.1336).

```

6.43.3.3 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller::looseMarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller`
(p. 2184).

```

6.43.3.4 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]

```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller`
(p. 2184).

```

6.43.3.5 virtual int ac-
tivemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException
) [virtual]

```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2185).

```
6.43.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2186).

```
6.43.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2186).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQMessageMarshaller.h`

6.44 `activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMa` Class Reference

Marshaling code for Open Wire Format for `ActiveMQMessageMarshaller` (p. 315).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQMessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller`:

Public Member Functions

- `ActiveMQMessageMarshaller ()`
- `virtual ~ActiveMQMessageMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaler.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write a object instance to data output stream.
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)`

Un-marshall an object instance from the data input stream.

- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)`

Write a object instance to data output stream.

6.44.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 315). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.44.2 Constructor & Destructor Documentation

6.44.2.1 `activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller::ActiveMQMessageMarshaller () [inline]`

6.44.2.2 `virtual activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller::~ActiveMQMessageMarshaller () [inline, virtual]`

6.44.3 Member Function Documentation

6.44.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1331).

6.44.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1336).

6.44.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2172).

6.44.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2172).

6.44.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2173).

```
6.44.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2174).

```
6.44.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2174).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQMessageMarshaller.h`

6.45 `activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMa` Class Reference

Marshaling code for Open Wire Format for `ActiveMQMessageMarshaller` (p. 319).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQMessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller`:

Public Member Functions

- `ActiveMQMessageMarshaller ()`
- `virtual ~ActiveMQMessageMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaler.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write a object instance to data output stream.
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)`

Un-marshall an object instance from the data input stream.

- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)`

Write a object instance to data output stream.

6.45.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 319). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.45.2 Constructor & Destructor Documentation

6.45.2.1 `activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller::ActiveMQMessageMarshaller () [inline]`

6.45.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller::~ActiveMQMessageMarshaller () [inline, virtual]`

6.45.3 Member Function Documentation

6.45.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.45.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.45.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2180).

6.45.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2180).

6.45.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2181).

```
6.45.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2182).

```
6.45.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2182).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQMessageMarshaller.h`

6.46 `activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMa` Class Reference

Marshaling code for Open Wire Format for `ActiveMQMessageMarshaller` (p. 323).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQMessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller`:

Public Member Functions

- `ActiveMQMessageMarshaller ()`
- `virtual ~ActiveMQMessageMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaler.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write a object instance to data output stream.
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)`

Un-marshall an object instance from the data input stream.

- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)`

Write a object instance to data output stream.

6.46.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 323). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.46.2 Constructor & Destructor Documentation

6.46.2.1 `activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller::ActiveMQMessageMarshaller () [inline]`

6.46.2.2 `virtual activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller::~ActiveMQMessageMarshaller () [inline, virtual]`

6.46.3 Member Function Documentation

6.46.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1331).

6.46.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1336).

6.46.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2167).

6.46.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2168).

6.46.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2169).

```
6.46.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2169).

```
6.46.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2170).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**ActiveMQMessageMarshaller.h**

6.47 activemq::commands::ActiveMQMessageTemplate< T > Class Template Reference

```
#include <src/main/activemq/commands/ActiveMQMessageTemplate.h>
```

Inheritance diagram for **activemq::commands::ActiveMQMessageTemplate< T >**:

Public Member Functions

- **ActiveMQMessageTemplate** ()
- virtual **~ActiveMQMessageTemplate** ()
- virtual void **acknowledge** () const throw (cms::IllegalStateException, cms::CMSException)
Acknowledges all consumed messages of the session of this consumed message.
- virtual void **onSend** ()
*Resets the **Message** (p. 2018) to a Read-Only state.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent.*
- virtual void **clearBody** () throw (cms::CMSException)
Clears out the body of the message.
- virtual void **clearProperties** () throw (cms::CMSException)
Clears the message properties.
- virtual std::vector< std::string > **getPropertyNames** () const throw (cms::CMSException)
Retrieves the property names.
- virtual bool **propertyExists** (const std::string &name) const throw (cms::CMSException)
Indicates whether or not a given property exists.
- virtual bool **getBooleanProperty** (const std::string &name) const throw (cms::MessageFormatException, cms::CMSException)

Gets a boolean property.

- virtual unsigned char **getByteProperty** (const std::string &name) const throw (cms::MessageFormatException, cms::CMSEException)

Gets a byte property.

- virtual double **getDoubleProperty** (const std::string &name) const throw (cms::MessageFormatException, cms::CMSEException)

Gets a double property.

- virtual float **getFloatProperty** (const std::string &name) const throw (cms::MessageFormatException, cms::CMSEException)

Gets a float property.

- virtual int **getIntProperty** (const std::string &name) const throw (cms::MessageFormatException, cms::CMSEException)

Gets a int property.

- virtual long long **getLongProperty** (const std::string &name) const throw (cms::MessageFormatException, cms::CMSEException)

Gets a long property.

- virtual short **getShortProperty** (const std::string &name) const throw (cms::MessageFormatException, cms::CMSEException)

Gets a short property.

- virtual std::string **getStringProperty** (const std::string &name) const throw (cms::MessageFormatException, cms::CMSEException)

Gets a string property.

- virtual void **setBooleanProperty** (const std::string &name, bool value) throw (cms::MessageNot WriteableException, cms::CMSEException)

Sets a boolean property.

- virtual void **setByteProperty** (const std::string &name, unsigned char value) throw (cms::MessageNot WriteableException, cms::CMSEException)

Sets a byte property.

- virtual void **setDoubleProperty** (const std::string &name, double value) throw (cms::MessageNot WriteableException, cms::CMSEException)

Sets a double property.

- virtual void **setFloatProperty** (const std::string &name, float value) throw (cms::MessageNot WriteableException, cms::CMSEException)

Sets a float property.

- virtual void **setIntProperty** (const std::string &name, int value) throw (cms::MessageNot WriteableException, cms::CMSEException)

Sets a int property.

- virtual void **setLongProperty** (const std::string &name, long long value) throw (cms::MessageNotWriteableException, cms::CMSException)
Sets a long property.
- virtual void **setShortProperty** (const std::string &name, short value) throw (cms::MessageNotWriteableException, cms::CMSException)
Sets a short property.
- virtual void **setStringProperty** (const std::string &name, const std::string &value) throw (cms::MessageNotWriteableException, cms::CMSException)
Sets a string property.
- virtual std::string **getCMSCorrelationID** () const throw (cms::CMSException)
Get the Correlation Id for this message.
- virtual void **setCMSCorrelationID** (const std::string &correlationId) throw (cms::CMSException)
Sets the Correlation Id used by this message.
- virtual int **getCMSDeliveryMode** () const throw (cms::CMSException)
Gets the DeliveryMode for this message.
- virtual void **setCMSDeliveryMode** (int mode) throw (cms::CMSException)
Sets the DeliveryMode for this message.
- virtual const cms::Destination * **getCMSDestination** () const throw (cms::CMSException)
*Gets the Destination for this **Message** (p. 2018), returns a.*
- virtual void **setCMSDestination** (const cms::Destination *destination) throw (cms::CMSException)
Sets the Destination for this message.
- virtual long long **getCMSExpiration** () const throw (cms::CMSException)
*Gets the Expiration Time for this **Message** (p. 2018).*
- virtual void **setCMSExpiration** (long long expireTime) throw (cms::CMSException)
Sets the Expiration Time for this message.
- virtual std::string **getCMSMessageID** () const throw (cms::CMSException)
*Gets the CMS **Message** (p. 2018) Id for this **Message** (p. 2018).*
- virtual void **setCMSMessageID** (const std::string &id AMQCPP_UNUSED) throw (cms::CMSException)
*Sets the CMS **Message** (p. 2018) Id for this message.*
- virtual int **getCMSPriority** () const throw (cms::CMSException)
*Gets the Priority Value for this **Message** (p. 2018).*
- virtual void **setCMSPriority** (int priority) throw (cms::CMSException)

Sets the Priority Value for this message.

- virtual bool **getCMSRedelivered** () const throw (cms::CMSEException)

*Gets the Redelivered Flag for this **Message** (p. 2018).*

- virtual void **setCMSRedelivered** (bool redelivered AMQCPP_UNUSED) throw (cms::CMSEException)

Sets the Redelivered Flag for this message.

- virtual const cms::Destination * **getCMSReplyTo** () const throw (cms::CMSEException)

*Gets the CMS Reply To Address for this **Message** (p. 2018).*

- virtual void **setCMSReplyTo** (const cms::Destination *destination) throw (cms::CMSEException)

Sets the CMS Reply To Address for this message.

- virtual long long **getCMSTimestamp** () const throw (cms::CMSEException)

*Gets the Time Stamp for this **Message** (p. 2018).*

- virtual void **setCMSTimestamp** (long long timeStamp) throw (cms::CMSEException)

Sets the Time Stamp for this message.

- virtual std::string **getCMSType** () const throw (cms::CMSEException)

*Gets the CMS **Message** (p. 2018) Type for this **Message** (p. 2018).*

- virtual void **setCMSType** (const std::string &type) throw (cms::CMSEException)

*Sets the CMS **Message** (p. 2018) Type for this message.*

Protected Member Functions

- void **failIfWriteOnlyBody** () const
- void **failIfReadOnlyBody** () const
- void **failIfReadOnlyProperties** () const

```
template<typename T> class activemq::commands::ActiveMQMessageTemplate< T
>
```

6.47.1 Constructor & Destructor Documentation

6.47.1.1 `template<typename T>
activemq::commands::ActiveMQMessageTemplate< T
>::ActiveMQMessageTemplate () [inline]`

6.47.1.2 `template<typename T> virtual
activemq::commands::ActiveMQMessageTemplate< T
>::~~ActiveMQMessageTemplate () [inline, virtual]`

6.47.2 Member Function Documentation

6.47.2.1 `template<typename T> virtual void
activemq::commands::ActiveMQMessageTemplate< T
>::acknowledge () const throw (cms::IllegalStateException,
cms::CMSException) [inline, virtual]`

Acknowledges all consumed messages of the session of this consumed message.

6.47.2.2 `template<typename T> virtual void
activemq::commands::ActiveMQMessageTemplate< T
>::clearBody () throw (cms::CMSException) [inline, virtual]`

Clears out the body of the message.

This does not clear the headers or properties.

Reimplemented in `activemq::commands::ActiveMQBytesMessage` (p. 170), `activemq::commands::ActiveMQMapMessage` (p. 275), `activemq::commands::ActiveMQStreamMessage` (p. 425), and `activemq::commands::ActiveMQTextMessage` (p. 527).

6.47.2.3 `template<typename T> virtual void
activemq::commands::ActiveMQMessageTemplate< T
>::clearProperties () throw (cms::CMSException) [inline, virtual]`

Clears the message properties.

Does not clear the body or header values.

6.47.2.4 `template<typename T> virtual bool
activemq::commands::ActiveMQMessageTemplate< T
>::equals (const DataStructure * value) const [inline, virtual]`

Compares the `DataStructure` (p. 1372) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::Message` (p. 2023).

Reimplemented in `activemq::commands::ActiveMQBlobMessage` (p. 144), `activemq::commands::ActiveMQBytesMessage` (p. 171), `activemq::commands::ActiveMQMapMessage` (p. 276), `activemq::commands::ActiveMQMessage` (p. 306), `activemq::commands::ActiveMQObjectMessage` (p. 346), `activemq::commands::ActiveMQStreamMessage` (p. 426), and `activemq::commands::ActiveMQTextMessage` (p. 527).

- 6.47.2.5** `template<typename T> void
activemq::commands::ActiveMQMessageTemplate< T
>::failIfReadOnlyBody () const [inline, protected]`
- 6.47.2.6** `template<typename T> void
activemq::commands::ActiveMQMessageTemplate< T
>::failIfReadOnlyProperties () const [inline, protected]`
- 6.47.2.7** `template<typename T> void
activemq::commands::ActiveMQMessageTemplate< T
>::failIfWriteOnlyBody () const [inline, protected]`
- 6.47.2.8** `template<typename T> virtual bool
activemq::commands::ActiveMQMessageTemplate< T
>::getBooleanProperty (const std::string & name) const throw (cms::MessageFormatException, cms::CMSException) [inline, virtual]`

Gets a boolean property.

Parameters

name The name of the property to retrieve.

Returns

The value for the named property.

Exceptions

CMSException if the property does not exist.

MessageFormatException - if this type conversion is invalid.

- 6.47.2.9** `template<typename T> virtual unsigned char
activemq::commands::ActiveMQMessageTemplate< T >::getByteProperty
(const std::string & name) const throw (cms::MessageFormatException,
cms::CMSException) [inline, virtual]`

Gets a byte property.

Parameters

name The name of the property to retrieve.

Returns

The value for the named property.

Exceptions

CMSEException if the property does not exist.

MessageFormatException - if this type conversion is invalid.

6.47.2.10 `template<typename T> virtual std::string
activemq::commands::ActiveMQMessageTemplate< T
>::getCMSCorrelationID () const throw (cms::CMSEException)
[inline, virtual]`

Get the Correlation Id for this message.

Returns

string representation of the correlation Id

Exceptions

CMSEException

6.47.2.11 `template<typename T> virtual int
activemq::commands::ActiveMQMessageTemplate< T
>::getCMSDeliveryMode () const throw (cms::CMSEException)
[inline, virtual]`

Gets the DeliveryMode for this message.

Returns

DeliveryMode enumerated value.

Exceptions

CMSEException

6.47.2.12 `template<typename T> virtual const cms::Destination*
activemq::commands::ActiveMQMessageTemplate< T
>::getCMSDestination () const throw (cms::CMSEException)
[inline, virtual]`

Gets the Destination for this **Message** (p. 2018), returns a.

Returns

Destination object

Exceptions

CMSEException

6.47.2.13 `template<typename T> virtual long long
activemq::commands::ActiveMQMessageTemplate< T
>::getCMSExpiration () const throw (cms::CMSException) [inline,
virtual]`

Gets the Expiration Time for this **Message** (p. 2018).

Returns

time value

Exceptions

CMSException

6.47.2.14 `template<typename T> virtual std::string
activemq::commands::ActiveMQMessageTemplate< T
>::getCMSMessageID () const throw (cms::CMSException)
[inline, virtual]`

Gets the CMS **Message** (p. 2018) Id for this **Message** (p. 2018).

Returns

time value

Exceptions

CMSException

6.47.2.15 `template<typename T> virtual int
activemq::commands::ActiveMQMessageTemplate< T
>::getCMSPriority () const throw (cms::CMSException) [inline,
virtual]`

Gets the Priority Value for this **Message** (p. 2018).

Returns

priority value

Exceptions

CMSException

6.47.2.16 `template<typename T> virtual bool
activemq::commands::ActiveMQMessageTemplate< T
>::getCMSRedelivered () const throw (cms::CMSException)
[inline, virtual]`

Gets the Redelivered Flag for this **Message** (p. 2018).

Returns

redelivered value

Exceptions

CMSException

6.47.2.17 `template<typename T> virtual const cms::Destination*
activemq::commands::ActiveMQMessageTemplate< T
>::getCMSReplyTo () const throw (cms::CMSException) [inline,
virtual]`

Gets the CMS Reply To Address for this **Message** (p.2018).

Returns

Reply To Value

Exceptions

CMSException

6.47.2.18 `template<typename T> virtual long long
activemq::commands::ActiveMQMessageTemplate< T
>::getCMSTimestamp () const throw (cms::CMSException)
[inline, virtual]`

Gets the Time Stamp for this **Message** (p.2018).

Returns

time stamp value

Exceptions

CMSException

6.47.2.19 `template<typename T> virtual std::string
activemq::commands::ActiveMQMessageTemplate< T
>::getCMSType () const throw (cms::CMSException) [inline,
virtual]`

Gets the CMS **Message** (p.2018) Type for this **Message** (p.2018).

Returns

type value

Exceptions

CMSException

```
6.47.2.20  template<typename T> virtual double
           activemq::commands::ActiveMQMessageTemplate< T
           >::getDoubleProperty ( const std::string &  name ) const throw (
           cms::MessageFormatException, cms::CMSException ) [inline, virtual]
```

Gets a double property.

Parameters

name The name of the property to retrieve.

Returns

The value for the named property.

Exceptions

CMSException if the property does not exist.

MessageFormatException - if this type conversion is invalid.

```
6.47.2.21  template<typename T> virtual float
           activemq::commands::ActiveMQMessageTemplate< T
           >::getFloatProperty ( const std::string &  name ) const throw (
           cms::MessageFormatException, cms::CMSException ) [inline, virtual]
```

Gets a float property.

Parameters

name The name of the property to retrieve.

Returns

The value for the named property.

Exceptions

CMSException if the property does not exist.

MessageFormatException - if this type conversion is invalid.

```
6.47.2.22  template<typename T> virtual int
           activemq::commands::ActiveMQMessageTemplate< T
           >::getIntProperty ( const std::string &  name ) const throw (
           cms::MessageFormatException, cms::CMSException ) [inline, virtual]
```

Gets a int property.

Parameters

name The name of the property to retrieve.

Returns

The value for the named property.

Exceptions

CMSEException if the property does not exist.

MessageFormatException - if this type conversion is invalid.

```
6.47.2.23  template<typename T> virtual long long
           activemq::commands::ActiveMQMessageTemplate< T
           >::getLongProperty ( const std::string & name ) const throw (
           cms::MessageFormatException, cms::CMSEException ) [inline, virtual]
```

Gets a long property.

Parameters

name The name of the property to retrieve.

Returns

The value for the named property.

Exceptions

CMSEException if the property does not exist.

MessageFormatException - if this type conversion is invalid.

```
6.47.2.24  template<typename T> virtual std::vector<std::string>
           activemq::commands::ActiveMQMessageTemplate< T
           >::getPropertyNames ( ) const throw ( cms::CMSEException ) [inline,
           virtual]
```

Retrieves the property names.

Returns

The complete set of property names currently in this message.

```
6.47.2.25  template<typename T> virtual short
           activemq::commands::ActiveMQMessageTemplate< T
           >::getShortProperty ( const std::string & name ) const throw (
           cms::MessageFormatException, cms::CMSEException ) [inline, virtual]
```

Gets a short property.

Parameters

name The name of the property to retrieve.

Returns

The value for the named property.

Exceptions

CMSEException if the property does not exist.

MessageFormatException - if this type conversion is invalid.

6.47.2.26 `template<typename T> virtual std::string
activemq::commands::ActiveMQMessageTemplate< T
>::getStringProperty (const std::string & name) const throw (
cms::MessageFormatException, cms::CMSException) [inline, virtual]`

Gets a string property.

Parameters

name The name of the property to retrieve.

Returns

The value for the named property.

Exceptions

CMSException if the property does not exist.

MessageFormatException - if this type conversion is invalid.

6.47.2.27 `template<typename T> virtual void
activemq::commands::ActiveMQMessageTemplate< T
>::onSend () [inline, virtual]`

Resets the **Message** (p. 2018) to a Read-Only state.

Reimplemented from **activemq::commands::Message** (p. 2030).

Reimplemented in **activemq::commands::ActiveMQBytesMessage** (p. 172), and **activemq::commands::ActiveMQStreamMessage** (p. 426).

6.47.2.28 `template<typename T> virtual bool
activemq::commands::ActiveMQMessageTemplate< T
>::propertyExists (const std::string & name) const throw (
cms::CMSException) [inline, virtual]`

Indicates whether or not a given property exists.

Parameters

name The name of the property to look up.

Returns

True if the property exists in this message.

6.47.2.29 `template<typename T> virtual void
activemq::commands::ActiveMQMessageTemplate< T
>::setBooleanProperty (const std::string & name, bool value) throw (
cms::MessageNotWriteableException, cms::CMSException) [inline,
virtual]`

Sets a boolean property.

Parameters

name The name of the property to retrieve.

value The value for the named property.

Exceptions

CMSException - if the name is an empty string.

MessageNotWriteableException - if properties are read-only

```
6.47.2.30  template<typename T> virtual void
           activemq::commands::ActiveMQMessageTemplate< T
           >::setByteProperty ( const std::string & name, unsigned char value )
           throw ( cms::MessageNotWriteableException, cms::CMSException )
           [inline, virtual]
```

Sets a byte property.

Parameters

name The name of the property to retrieve.

value The value for the named property.

Exceptions

CMSException - if the name is an empty string.

MessageNotWriteableException - if properties are read-only

```
6.47.2.31  template<typename T> virtual void
           activemq::commands::ActiveMQMessageTemplate< T
           >::setCMSCorrelationID ( const std::string & correlationId ) throw (
           cms::CMSException ) [inline, virtual]
```

Sets the Correlation Id used by this message.

Parameters

correlationId - String representing the correlation id.

Exceptions

CMSException

```
6.47.2.32  template<typename T> virtual void
           activemq::commands::ActiveMQMessageTemplate< T
           >::setCMSDeliveryMode ( int mode ) throw ( cms::CMSException )
           [inline, virtual]
```

Sets the DeliveryMode for this message.

Parameters

mode - DeliveryMode enumerated value.

Exceptions

CMSException

```
6.47.2.33  template<typename T> virtual void
             activemq::commands::ActiveMQMessageTemplate< T
             >::setCMSDestination ( const cms::Destination * destination ) throw (
             cms::CMSException ) [inline, virtual]
```

Sets the Destination for this message.

Parameters

destination - Destination Object

Exceptions

CMSException

```
6.47.2.34  template<typename T> virtual void
             activemq::commands::ActiveMQMessageTemplate< T
             >::setCMSExpiration ( long long expireTime ) throw (
             cms::CMSException ) [inline, virtual]
```

Sets the Expiration Time for this message.

Parameters

expireTime - time value

Exceptions

CMSException

```
6.47.2.35  template<typename T> virtual void
             activemq::commands::ActiveMQMessageTemplate< T
             >::setCMSMessageID ( const std::string &id AMQCPP_UNUSED )
             throw ( cms::CMSException ) [inline, virtual]
```

Sets the CMS **Message** (p. 2018) Id for this message.

Parameters

id - time value

Exceptions

CMSException

6.47.2.36 `template<typename T> virtual void
activemq::commands::ActiveMQMessageTemplate< T
>::setCMSPriority (int priority) throw (cms::CMSException)
[inline, virtual]`

Sets the Priority Value for this message.

Parameters

priority - priority value for this message

Exceptions

CMSException

6.47.2.37 `template<typename T> virtual void
activemq::commands::ActiveMQMessageTemplate< T
>::setCMSRedelivered (bool redelivered AMQCPP_UNUSED)
throw (cms::CMSException) [inline, virtual]`

Sets the Redelivered Flag for this message.

Parameters

redelivered - boolean redelivered value

Exceptions

CMSException

6.47.2.38 `template<typename T> virtual void
activemq::commands::ActiveMQMessageTemplate< T
>::setCMSReplyTo (const cms::Destination * destination) throw (cms::CMSException) [inline, virtual]`

Sets the CMS Reply To Address for this message.

Parameters

destination Pointer to the CMS Destination that is the Reply-To value.

Exceptions

CMSException

6.47.2.39 `template<typename T> virtual void
activemq::commands::ActiveMQMessageTemplate< T
>::setCMSTimestamp (long long timeStamp) throw (cms::CMSException) [inline, virtual]`

Sets the Time Stamp for this message.

Parameters

timeStamp - integer time stamp value

Exceptions

CMSException

```
6.47.2.40  template<typename T> virtual void
           activemq::commands::ActiveMQMessageTemplate< T
           >::setCMSType ( const std::string & type ) throw ( cms::CMSException
           ) [inline, virtual]
```

Sets the CMS **Message** (p. 2018) Type for this message.

Parameters

type - message type value string

Exceptions

CMSException

```
6.47.2.41  template<typename T> virtual void
           activemq::commands::ActiveMQMessageTemplate< T
           >::setDoubleProperty ( const std::string & name, double value )
           throw ( cms::MessageNotWriteableException, cms::CMSException )
           [inline, virtual]
```

Sets a double property.

Parameters

name The name of the property to retrieve.

value The value for the named property.

Exceptions

CMSException - if the name is an empty string.

MessageNotWriteableException - if properties are read-only

```
6.47.2.42  template<typename T> virtual void
           activemq::commands::ActiveMQMessageTemplate< T
           >::setFloatProperty ( const std::string & name, float value ) throw (
           cms::MessageNotWriteableException, cms::CMSException ) [inline,
           virtual]
```

Sets a float property.

Parameters

name The name of the property to retrieve.

value The value for the named property.

Exceptions

CMSException - if the name is an empty string.

MessageNotWriteableException - if properties are read-only

6.47.2.43 `template<typename T> virtual void
activemq::commands::ActiveMQMessageTemplate< T
>::setIntProperty (const std::string & name, int value) throw (cms::MessageNotWriteableException, cms::CMSException) [inline, virtual]`

Sets a int property.

Parameters

name The name of the property to retrieve.

value The value for the named property.

Exceptions

CMSException - if the name is an empty string.

MessageNotWriteableException - if properties are read-only

6.47.2.44 `template<typename T> virtual void
activemq::commands::ActiveMQMessageTemplate< T
>::setLongProperty (const std::string & name, long long value)
throw (cms::MessageNotWriteableException, cms::CMSException)
[inline, virtual]`

Sets a long property.

Parameters

name The name of the property to retrieve.

value The value for the named property.

Exceptions

CMSException - if the name is an empty string.

MessageNotWriteableException - if properties are read-only

6.47.2.45 `template<typename T> virtual void
activemq::commands::ActiveMQMessageTemplate< T
>::setShortProperty (const std::string & name, short value) throw (cms::MessageNotWriteableException, cms::CMSException) [inline, virtual]`

Sets a short property.

Parameters

name The name of the property to retrieve.

value The value for the named property.

Exceptions

CMSEException - if the name is an empty string.

MessageNotWriteableException - if properties are read-only

```
6.47.2.46  template<typename T> virtual void
            activemq::commands::ActiveMQMessageTemplate< T
            >::setStringProperty (  const std::string &  name,  const
            std::string &  value  ) throw ( cms::MessageNotWriteableException,
            cms::CMSEException ) [inline, virtual]
```

Sets a string property.

Parameters

name The name of the property to retrieve.

value The value for the named property.

Exceptions

CMSEException - if the name is an empty string.

MessageNotWriteableException - if properties are read-only

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**ActiveMQMessageTemplate.h**

6.48 activemq::commands::ActiveMQObjectMessage Class Reference

```
#include <src/main/activemq/commands/ActiveMQObjectMessage.h>
```

Inheritance diagram for activemq::commands::ActiveMQObjectMessage:

Public Member Functions

- **ActiveMQObjectMessage** ()
- virtual **~ActiveMQObjectMessage** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **ActiveMQObjectMessage * cloneDataStructure** () const

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1372) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent.*
- virtual **cms::Message** * **clone** () const
Clone this message exactly, returns a new instance that the caller is required to delete.

Static Public Attributes

- static const unsigned char **ID_ACTIVEMQOBJECTMESSAGE** = 26

6.48.1 Constructor & Destructor Documentation

6.48.1.1 **activemq::commands::ActiveMQObjectMessage::ActiveMQObjectMessage**
()

6.48.1.2 **virtual**
activemq::commands::ActiveMQObjectMessage::~~ActiveMQObjectMessage
() [inline, virtual]

6.48.2 Member Function Documentation

6.48.2.1 **virtual cms::Message*** **activemq::commands::ActiveMQObjectMessage::clone** ()
const [inline, virtual]

Clone this message exactly, returns a new instance that the caller is required to delete.

Returns

new copy of this message

6.48.2.2 **virtual ActiveMQObjectMessage*** **activemq::commands::ActiveMQObjectMessage::cloneDataStructure** ()
const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Reimplemented from `activemq::commands::Message` (p. 2023).

6.48.2.3 `virtual void activemq::commands::ActiveMQObjectMessage::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

`src` - Source Object

Reimplemented from `activemq::commands::Message` (p. 2023).

6.48.2.4 `virtual bool activemq::commands::ActiveMQObjectMessage::equals (const DataStructure * value) const [virtual]`

Compares the `DataStructure` (p. 1372) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if `DataStructure`'s are Equal.

Reimplemented from `activemq::commands::ActiveMQMessageTemplate<cms::ObjectMessage>` (p. 331).

6.48.2.5 `virtual unsigned char activemq::commands::ActiveMQObjectMessage::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

Returns

new `DataStructure` (p. 1372) type copy.

Reimplemented from `activemq::commands::Message` (p. 2025).

6.48.2.6 `virtual std::string activemq::commands::ActiveMQObjectMessage::toString () const [virtual]`

Returns a string containing the information for this `DataStructure` (p. 1372) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from `activemq::commands::Message` (p. 2033).

6.48.3 Field Documentation

- 6.48.3.1 `const unsigned char activemq::commands::ActiveMQObjectMessage::ID_ - ACTIVEMQOBJECTMESSAGE = 26` [static]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ActiveMQObjectMessage.h`

6.49 activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller Class Reference

Marshaling code for Open Wire Format for `ActiveMQObjectMessageMarshaller` (p. 347).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQObjectMessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller`:

Public Member Functions

- `ActiveMQObjectMessageMarshaller ()`
- `virtual ~ActiveMQObjectMessageMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaler.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshall an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write a object instance to data output stream.
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)`
Un-marshall an object instance from the data input stream.

- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)`

Write a object instance to data output stream.

6.49.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 347).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.49.2 Constructor & Destructor Documentation

6.49.2.1 `activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller::ActiveMQObjectMessageMarshaller () [inline]`

6.49.2.2 `virtual activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller::~ActiveMQObjectMessageMarshaller () [inline, virtual]`

6.49.3 Member Function Documentation

6.49.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1331).

6.49.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1336).

6.49

activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller

Class Reference

351

```
6.49.3.3 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller::looseMarshal(
    ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
    decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2176).

```
6.49.3.4 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller::looseUnmarshal(
    ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn ) throw (
    decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2176).

```
6.49.3.5 virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller::tightMarshal(
    ( OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException
    ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2177).

```
6.49.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller::tightMarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2178).

```
6.49.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller::tightUnma
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2178).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQObjectMessageMarshaller.h`

6.50 activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller Class Reference

Marshaling code for Open Wire Format for `ActiveMQObjectMessageMarshaller` (p. 351).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQObjectMessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller`:

Public Member Functions

- `ActiveMQObjectMessageMarshaller ()`
- `virtual ~ActiveMQObjectMessageMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaller.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write a object instance to data output stream.
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)`

Un-marshall an object instance from the data input stream.

- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)`

Write a object instance to data output stream.

6.50.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 351).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.50.2 Constructor & Destructor Documentation

6.50.2.1 `activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller::ActiveMQObjectMessageMarshaller () [inline]`

6.50.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller::~ActiveMQObjectMessageMarshaller () [inline, virtual]`

6.50.3 Member Function Documentation

6.50.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1331).

6.50.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1336).

6.50

activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller

Class Reference

355

```
6.50.3.3 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller::looseMars
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller`
(p. 2184).

```
6.50.3.4 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller::looseUnmars
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller`
(p. 2184).

```
6.50.3.5 virtual int ac-
tivemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller::tightMars
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException
) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2185).

```
6.50.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller::tightMarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2186).

```
6.50.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller::tightUnma
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2186).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQObjectMessageMarshaller.h`

6.51 activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller Class Reference

Marshaling code for Open Wire Format for `ActiveMQObjectMessageMarshaller` (p. 355).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQObjectMessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller`:

Public Member Functions

- `ActiveMQObjectMessageMarshaller ()`
- `virtual ~ActiveMQObjectMessageMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaller.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write a object instance to data output stream.
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)`

Un-marshall an object instance from the data input stream.

- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)`

Write a object instance to data output stream.

6.51.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 355).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.51.2 Constructor & Destructor Documentation

6.51.2.1 `activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller::ActiveMQObjectMessageMarshaller () [inline]`

6.51.2.2 `virtual activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller::~ActiveMQObjectMessageMarshaller () [inline, virtual]`

6.51.3 Member Function Documentation

6.51.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1331).

6.51.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1336).

```

6.51.3.3 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller::looseMars
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2172).

```

6.51.3.4 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller::looseUnmars
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]

```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2172).

```

6.51.3.5 virtual int ac-
tivemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller::tightMars
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException
) [virtual]

```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2173).

```
6.51.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller::tightMarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2174).

```
6.51.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller::tightUnma
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2174).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQObjectMessageMarshaller.h`

6.52 activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller Class Reference

Marshaling code for Open Wire Format for `ActiveMQObjectMessageMarshaller` (p. 359).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQObjectMessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller`:

Public Member Functions

- `ActiveMQObjectMessageMarshaller ()`
- `virtual ~ActiveMQObjectMessageMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaller.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write a object instance to data output stream.
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)`

Un-marshall an object instance from the data input stream.

- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)`

Write a object instance to data output stream.

6.52.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 359).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.52.2 Constructor & Destructor Documentation

6.52.2.1 `activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller::ActiveMQObjectMessageMarshaller () [inline]`

6.52.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller::~ActiveMQObjectMessageMarshaller () [inline, virtual]`

6.52.3 Member Function Documentation

6.52.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.52.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

```

6.52.3.3 virtual void ac-
        tivemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller::looseMars
        ( OpenWireFormat * wireFormat, commands::DataStructure *
          dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
          decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2180).

```

6.52.3.4 virtual void ac-
        tivemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller::looseUnm
        ( OpenWireFormat * wireFormat, commands::DataStructure *
          dataStructure, decaf::io::DataInputStream * dataIn ) throw (
          decaf::io::IOException ) [virtual]

```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2180).

```

6.52.3.5 virtual int ac-
        tivemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller::tightMars
        ( OpenWireFormat * wireFormat, commands::DataStructure *
          dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException
        ) [virtual]

```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2181).

```
6.52.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller::tightMarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2182).

```
6.52.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller::tightUnma
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2182).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQObjectMessageMarshaller.h`

6.53 activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller Class Reference

Marshaling code for Open Wire Format for `ActiveMQObjectMessageMarshaller` (p. 363).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQObjectMessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller`:

Public Member Functions

- `ActiveMQObjectMessageMarshaller ()`
- `virtual ~ActiveMQObjectMessageMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaller.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write a object instance to data output stream.
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)`

Un-marshall an object instance from the data input stream.

- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)`

Write a object instance to data output stream.

6.53.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 363).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.53.2 Constructor & Destructor Documentation

6.53.2.1 `activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller::ActiveMQObjectMessageMarshaller () [inline]`

6.53.2.2 `virtual activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller::~ActiveMQObjectMessageMarshaller () [inline, virtual]`

6.53.3 Member Function Documentation

6.53.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.53.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

```

6.53.3.3 virtual void ac-
          tivemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller::looseMars
          ( OpenWireFormat * wireFormat, commands::DataStructure *
            dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
            decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2167).

```

6.53.3.4 virtual void ac-
          tivemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller::looseUnmars
          ( OpenWireFormat * wireFormat, commands::DataStructure *
            dataStructure, decaf::io::DataInputStream * dataIn ) throw (
            decaf::io::IOException ) [virtual]

```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2168).

```

6.53.3.5 virtual int ac-
          tivemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller::tightMars
          ( OpenWireFormat * wireFormat, commands::DataStructure *
            dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException
            ) [virtual]

```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2169).

```
6.53.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller::tightMarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2169).

```
6.53.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller::tightUnma
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 2170).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQObjectMessageMarshaller.h`

6.54 activemq::core::ActiveMQProducer Class Reference

```
#include <src/main/activemq/core/ActiveMQProducer.h>
```

Inheritance diagram for **activemq::core::ActiveMQProducer**:

Public Member Functions

- **ActiveMQProducer** (const **Pointer**< **commands::ProducerInfo** > &producerInfo, const **Pointer**< **cms::Destination** > &destination, **ActiveMQSession** *session)

*Constructor, creates an instance of an **ActiveMQProducer** (p. 367).*

- virtual **~ActiveMQProducer** ()
- virtual void **close** () throw (cms::CMSException)

Closes the Consumer.

- virtual void **send** (cms::Message *message) throw (cms::CMSException, cms::MessageFormatException, cms::InvalidDestinationException, cms::UnsupportedOperationException)

Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.

- virtual void **send** (cms::Message *message, int deliveryMode, int priority, long long timeToLive) throw (cms::CMSException, cms::MessageFormatException, cms::InvalidDestinationException, cms::UnsupportedOperationException)

Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.

- virtual void **send** (const cms::Destination *destination, cms::Message *message) throw (cms::CMSException, cms::MessageFormatException, cms::InvalidDestinationException, cms::UnsupportedOperationException)

Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.

- virtual void **send** (const cms::Destination *destination, cms::Message *message, int deliveryMode, int priority, long long timeToLive) throw (cms::CMSException, cms::MessageFormatException, cms::InvalidDestinationException, cms::UnsupportedOperationException)

Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.

- virtual void **setDeliveryMode** (int mode) throw (cms::CMSEException)
Sets the delivery mode for this Producer.
- virtual int **getDeliveryMode** () const throw (cms::CMSEException)
Gets the delivery mode for this Producer.
- virtual void **setDisableMessageID** (bool value) throw (cms::CMSEException)
Sets if Message Ids are disabled for this Producer.
- virtual bool **getDisableMessageID** () const throw (cms::CMSEException)
Gets if Message Ids are disabled for this Producer.
- virtual void **setDisableMessageTimeStamp** (bool value) throw (cms::CMSEException)
Sets if Message Time Stamps are disabled for this Producer.
- virtual bool **getDisableMessageTimeStamp** () const throw (cms::CMSEException)
Gets if Message Time Stamps are disabled for this Producer.
- virtual void **setPriority** (int priority) throw (cms::CMSEException)
Sets the Priority that this Producers sends messages at.
- virtual int **getPriority** () const throw (cms::CMSEException)
Gets the Priority level that this producer sends messages at.
- virtual void **setTimeToLive** (long long time) throw (cms::CMSEException)
Sets the Time to Live that this Producers sends messages with.
- virtual long long **getTimeToLive** () const throw (cms::CMSEException)
Gets the Time to Live that this producer sends messages with.
- virtual void **setSendTimeout** (long long time) throw (cms::CMSEException)
Sets the Send Timeout that this Producers sends messages with.
- virtual long long **getSendTimeout** () const throw (cms::CMSEException)
Gets the Send Timeout that this producer sends messages with.
- bool **isClosed** () const
- const **commands::ProducerInfo** & **getProducerInfo** () const
Retries this object ProducerInfo pointer.
- **commands::ProducerId** & **getProducerId** () const
Retries this object ProducerId or NULL if closed.
- virtual void **onProducerAck** (const **commands::ProducerAck** &ack)
Handles the work of Processing a ProducerAck Command from the Broker.

6.54.1 Constructor & Destructor Documentation

6.54.1.1 `activemq::core::ActiveMQProducer::ActiveMQProducer (const Pointer< commands::ProducerInfo > & producerInfo, const Pointer< cms::Destination > & destination, ActiveMQSession * session)`

Constructor, creates an instance of an **ActiveMQProducer** (p. 367).

Parameters

producerInfo Pointer to a ProducerInfo command which identifies this producer.
destination The assigned Destination this Producer sends to, or null if not set. The Producer does not own the Pointer passed.
session The Session which is the parent of this Producer.

6.54.1.2 `virtual activemq::core::ActiveMQProducer::~~ActiveMQProducer ()`
 [virtual]

6.54.2 Member Function Documentation

6.54.2.1 `virtual void activemq::core::ActiveMQProducer::close () throw (cms::CMSException)` [virtual]

Closes the Consumer.

This will return all allocated resources and purge any outstanding messages. This method will block if there is a call to receive in progress, or a dispatch to a MessageListener in place

Exceptions

CMSException

Implements **cms::Closeable** (p. 952).

6.54.2.2 `virtual int activemq::core::ActiveMQProducer::getDeliveryMode () const throw (cms::CMSException)` [inline, virtual]

Gets the delivery mode for this Producer.

Returns

The DeliveryMode

Implements **cms::MessageProducer** (p. 2191).

6.54.2.3 `virtual bool activemq::core::ActiveMQProducer::getDisableMessageID () const throw (cms::CMSException)` [inline, virtual]

Gets if Message Ids are disabled for this Producer.

Returns

a boolean indicating state enable / disable (true / false) for MessageIds.

Implements **cms::MessageProducer** (p. 2191).

6.54.2.4 `virtual bool activemq::core::ActiveMQProducer::getDisableMessageTimeStamp () const throw (cms::CMSException) [inline, virtual]`

Gets if Message Time Stamps are disabled for this Producer.

Returns

boolean indicating state of enable / disable (true / false)

Implements `cms::MessageProducer` (p. 2192).

6.54.2.5 `virtual int activemq::core::ActiveMQProducer::getPriority () const throw (cms::CMSException) [inline, virtual]`

Gets the Priority level that this producer sends messages at.

Returns

int based priority level

Implements `cms::MessageProducer` (p. 2192).

6.54.2.6 `commands::ProducerId& activemq::core::ActiveMQProducer::getProducerId () const [inline]`

Retries this object ProducerId or NULL if closed.

Returns

ProducerId Reference

6.54.2.7 `const commands::ProducerInfo& activemq::core::ActiveMQProducer::getProducerInfo () const [inline]`

Retries this object ProducerInfo pointer.

Returns

ProducerInfo Reference

6.54.2.8 `virtual long long activemq::core::ActiveMQProducer::getSendTimeout () const throw (cms::CMSException) [inline, virtual]`

Gets the Send Timeout that this producer sends messages with.

Returns

The default send timeout value in milliseconds.

6.54.2.9 `virtual long long activemq::core::ActiveMQProducer::getTimeToLive ()
const throw (cms::CMSException) [inline, virtual]`

Gets the Time to Live that this producer sends messages with.

Returns

The default time to live value in milliseconds.

Implements **cms::MessageProducer** (p. 2192).

6.54.2.10 `bool activemq::core::ActiveMQProducer::isClosed () const [inline]`

Returns

true if this Producer has been closed.

6.54.2.11 `virtual void activemq::core::ActiveMQProducer::onProducerAck (const
commands::ProducerAck & ack) [virtual]`

Handles the work of Processing a ProducerAck Command from the Broker.

Parameters

ack - The ProducerAck message received from the Broker.

6.54.2.12 `virtual void activemq::core::ActiveMQProducer::send (const
cms::Destination * destination, cms::Message * message,
int deliveryMode, int priority, long long timeToLive)
throw (cms::CMSException, cms::MessageFormatException,
cms::InvalidDestinationException, cms::UnsupportedOperationException
) [virtual]`

Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.

Parameters

destination The destination on which to send the message

message The message to be sent.

deliveryMode The delivery mode to be used.

priority The priority for this message.

timeToLive The time to live value for this message in milliseconds.

Exceptions

CMSException - if an internal error occurs while sending the message.

MessageFormatException - if an Invalid Message is given.

InvalidDestinationException - if a client uses this method with a MessageProducer with an invalid destination.

UnsupportedOperationException - if a client uses this method with a MessageProducer that did not specify a destination at creation time.

Implements **cms::MessageProducer** (p. 2193).

6.54.2.13 `virtual void activemq::core::ActiveMQProducer::send (cms::Message *
message) throw (cms::CMSException, cms::MessageFormatException,
cms::InvalidDestinationException, cms::UnsupportedOperationException
) [virtual]`

Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.

Uses default values for deliveryMode, priority, and time to live.

Parameters

message The message to be sent.

Exceptions

CMSException - if an internal error occurs while sending the message.

MessageFormatException - if an Invalid Message is given.

InvalidDestinationException - if a client uses this method with a MessageProducer with an invalid destination.

UnsupportedOperationException - if a client uses this method with a MessageProducer that did not specify a destination at creation time.

Implements **cms::MessageProducer** (p. 2194).

6.54.2.14 `virtual void activemq::core::ActiveMQProducer::send (cms::Message
* message, int deliveryMode, int priority, long long timeToLive
) throw (cms::CMSException, cms::MessageFormatException,
cms::InvalidDestinationException, cms::UnsupportedOperationException
) [virtual]`

Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.

Parameters

message The message to be sent.

deliveryMode The delivery mode to be used.

priority The priority for this message.

timeToLive The time to live value for this message in milliseconds.

Exceptions

CMSException - if an internal error occurs while sending the message.

MessageFormatException - if an Invalid Message is given.

InvalidDestinationException - if a client uses this method with a MessageProducer with an invalid destination.

UnsupportedOperationException - if a client uses this method with a MessageProducer that did not specify a destination at creation time.

Implements **cms::MessageProducer** (p. 2192).

6.54.2.15 `virtual void activemq::core::ActiveMQProducer::send (const cms::Destination * destination, cms::Message * message) throw (cms::CMSException, cms::MessageFormatException, cms::InvalidDestinationException, cms::UnsupportedOperationException) [virtual]`

Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.

Uses default values for deliveryMode, priority, and time to live.

Parameters

destination The destination on which to send the message

message the message to be sent.

Exceptions

CMSException - if an internal error occurs while sending the message.

MessageFormatException - if an Invalid Message is given.

InvalidDestinationException - if a client uses this method with a MessageProducer with an invalid destination.

UnsupportedOperationException - if a client uses this method with a MessageProducer that did not specify a destination at creation time.

Implements `cms::MessageProducer` (p. 2194).

6.54.2.16 `virtual void activemq::core::ActiveMQProducer::setDeliveryMode (int mode) throw (cms::CMSException) [inline, virtual]`

Sets the delivery mode for this Producer.

Parameters

mode - The DeliveryMode to use for Message sends.

Implements `cms::MessageProducer` (p. 2195).

6.54.2.17 `virtual void activemq::core::ActiveMQProducer::setDisableMessageID (bool value) throw (cms::CMSException) [inline, virtual]`

Sets if Message Ids are disabled for this Producer.

Parameters

value - boolean indicating enable / disable (true / false)

Implements `cms::MessageProducer` (p. 2195).

6.54.2.18 `virtual void activemq::core::ActiveMQProducer::setDisableMessageTimeStamp (bool value) throw (cms::CMSExcption) [inline, virtual]`

Sets if Message Time Stamps are disabled for this Producer.

Parameters

value - boolean indicating enable / disable (true / false)

Implements **cms::MessageProducer** (p. 2195).

6.54.2.19 `virtual void activemq::core::ActiveMQProducer::setPriority (int priority) throw (cms::CMSExcption) [inline, virtual]`

Sets the Priority that this Producers sends messages at.

Parameters

priority int value for Priority level

Implements **cms::MessageProducer** (p. 2195).

6.54.2.20 `virtual void activemq::core::ActiveMQProducer::setSendTimeout (long long time) throw (cms::CMSExcption) [inline, virtual]`

Sets the Send Timeout that this Producers sends messages with.

Parameters

time The new default send timeout value in milliseconds.

6.54.2.21 `virtual void activemq::core::ActiveMQProducer::setTimeToLive (long long time) throw (cms::CMSExcption) [inline, virtual]`

Sets the Time to Live that this Producers sends messages with.

Parameters

time The new default time to live value in milliseconds.

Implements **cms::MessageProducer** (p. 2196).

The documentation for this class was generated from the following file:

- `src/main/activemq/core/ActiveMQProducer.h`

6.55 activemq::util::ActiveMQProperties Class Reference

Implementation of the CMSProperties interface that delegates to a **decaf::util::Properties** (p. 2494) object.


```
#include <src/main/activemq/util/ActiveMQProperties.h>
```

Inheritance diagram for activemq::util::ActiveMQProperties:

Public Member Functions

- virtual `~ActiveMQProperties ()`
- virtual `decaf::util::Properties & getProperties ()`
- virtual const `decaf::util::Properties & getProperties () const`
- virtual void `setProperties (decaf::util::Properties &props)`
- virtual bool `isEmpty () const`
Returns true if the properties object is empty.
- virtual const char * `getProperty (const std::string &name) const`
Looks up the value for the given property.
- virtual std::string `getProperty (const std::string &name, const std::string &defaultValue) const`
Looks up the value for the given property.
- virtual void `setProperty (const std::string &name, const std::string &value)`
Sets the value for a given property.
- virtual bool `hasProperty (const std::string &name) const`
Check to see if the Property exists in the set.
- virtual void `remove (const std::string &name)`
Removes the property with the given name.
- virtual std::vector< std::pair< std::string, std::string > > `toArray () const`
Method that serializes the contents of the property map to an array.
- virtual void `copy (const CMSProperties *source)`
Copies the contents of the given properties object to this one.
- virtual CMSProperties * `clone () const`
Clones this object.
- virtual void `clear ()`
Clears all properties from the map.
- virtual std::string `toString () const`
Formats the contents of the Properties Object into a string that can be logged, etc.

6.55.1 Detailed Description

Implementation of the CMSProperties interface that delegates to a **decaf::util::Properties** (p. 2494) object.

Since

2.0

6.55.2 Constructor & Destructor Documentation

6.55.2.1 `virtual activemq::util::ActiveMQProperties::~ActiveMQProperties ()`
[virtual]

6.55.3 Member Function Documentation

6.55.3.1 `virtual void activemq::util::ActiveMQProperties::clear ()` [inline,
virtual]

Clears all properties from the map.

Implements **cms::CMSProperties** (p. 966).

6.55.3.2 `virtual CMSProperties* activemq::util::ActiveMQProperties::clone ()`
`const` [virtual]

Clones this object.

Returns

a replica of this object.

Implements **cms::CMSProperties** (p. 966).

6.55.3.3 `virtual void activemq::util::ActiveMQProperties::copy (const`
`CMSProperties * source)` [virtual]

Copies the contents of the given properties object to this one.

Parameters

source The source properties object.

6.55.3.4 `virtual decaf::util::Properties& activemq::util::ActiveMQProperties::getProperties ()`
[inline, virtual]

6.55.3.5 `virtual const decaf::util::Properties& activemq::util::ActiveMQProperties::getProperties ()`
`const` [inline, virtual]

6.55.3.6 `virtual const char* activemq::util::ActiveMQProperties::getProperty (`
`const std::string & name) const` [inline, virtual]

Looks up the value for the given property.

Parameters

name The name of the property to be looked up.

Returns

the value of the property with the given name, if it exists. If it does not exist, returns NULL.

Implements `cms::CMSProperties` (p. 966).

6.55.3.7 `virtual std::string activemq::util::ActiveMQProperties::getProperty (`
`const std::string & name, const std::string & defaultValue) const`
[inline, virtual]

Looks up the value for the given property.

Parameters

name the name of the property to be looked up.

defaultValue The value to be returned if the given property does not exist.

Returns

The value of the property specified by *name*, if it exists, otherwise the *defaultValue*.

Implements `cms::CMSProperties` (p. 967).

6.55.3.8 `virtual bool activemq::util::ActiveMQProperties::hasProperty (const`
`std::string & name) const` [inline, virtual]

Check to see if the Property exists in the set.

Parameters

name - property name to check for in this properties set.

Returns

true if property exists, false otherwise.

Implements `cms::CMSProperties` (p. 967).

6.55.3.9 `virtual bool activemq::util::ActiveMQProperties::isEmpty () const`
[inline, virtual]

Returns true if the properties object is empty.

Returns

true if empty

Implements `cms::CMSProperties` (p. 967).

6.55.3.10 `virtual void activemq::util::ActiveMQProperties::remove (const`
`std::string & name)` [inline, virtual]

Removes the property with the given name.

Parameters

name the name of the property to remove.

Implements `cms::CMSProperties` (p. 967).

6.55.3.11 `virtual void activemq::util::ActiveMQProperties::setProperties (`
`decaf::util::Properties & props)` [inline, virtual]

6.55.3.12 `virtual void activemq::util::ActiveMQProperties::setProperty (const`
`std::string & name, const std::string & value)` [inline, virtual]

Sets the value for a given property.

If the property already exists, overwrites the value.

Parameters

name The name of the value to be written.

value The value to be written.

Implements `cms::CMSProperties` (p. 968).

6.55.3.13 `virtual std::vector< std::pair< std::string, std::string > >`
`activemq::util::ActiveMQProperties::toArray () const` [inline,
virtual]

Method that serializes the contents of the property map to an array.

Returns

list of pairs where the first is the name and the second is the value.

Implements `cms::CMSProperties` (p. 968).

6.55.3.14 virtual std::string activemq::util::ActiveMQProperties::toString () const [inline, virtual]

Formats the contents of the Properties Object into a string that can be logged, etc.

Returns

string value of this object.

Implements **cms::CMSProperties** (p. 968).

The documentation for this class was generated from the following file:

- src/main/activemq/util/**ActiveMQProperties.h**

6.56 activemq::commands::ActiveMQQueue Class Reference

```
#include <src/main/activemq/commands/ActiveMQQueue.h>
```

Inheritance diagram for activemq::commands::ActiveMQQueue:

Public Member Functions

- **ActiveMQQueue** ()
- **ActiveMQQueue** (const std::string &name)
- virtual ~**ActiveMQQueue** ()
- virtual unsigned char **getDataStructureType** () const
- virtual **ActiveMQQueue** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1372) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent.*
- virtual const **cms::Destination** * **getCMSDestination** () const
- virtual **cms::Destination::DestinationType** **getDestinationType** () const
Retrieve the Destination Type for this Destination.
- virtual **cms::Destination** * **clone** () const
Creates a new instance of this destination type that is a copy of this one, and returns it.
- virtual void **copy** (const **cms::Destination** &source)

Copies the contents of the given Destination object to this one.

- virtual const **cms::CMSProperties** & **getCMSProperties** () const
Retrieve any properties that might be part of the destination that was specified.
- virtual std::string **getQueueName** () const throw (cms::CMSException)
Gets the name of this queue.

Static Public Attributes

- static const unsigned char **ID_ACTIVEMQQUEUE** = 100

6.56.1 Constructor & Destructor Documentation

- 6.56.1.1 **activemq::commands::ActiveMQQueue::ActiveMQQueue** ()
- 6.56.1.2 **activemq::commands::ActiveMQQueue::ActiveMQQueue** (const std::string & *name*)
- 6.56.1.3 **virtual activemq::commands::ActiveMQQueue::~~ActiveMQQueue** ()
[inline, virtual]

6.56.2 Member Function Documentation

- 6.56.2.1 **virtual cms::Destination* activemq::commands::ActiveMQQueue::clone** () const [inline, virtual]

Creates a new instance of this destination type that is a copy of this one, and returns it.

Returns

cloned copy of this object

Implements **cms::Destination** (p.1389).

- 6.56.2.2 **virtual ActiveMQQueue* activemq::commands::ActiveMQQueue::cloneDataStructure** () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

- 6.56.2.3 **virtual void activemq::commands::ActiveMQQueue::copy** (const cms::Destination & *source*) [inline, virtual]

Copies the contents of the given Destination object to this one.

Parameters

source The source Destination object.

Implements **cms::Destination** (p.1389).

6.56.2.4 virtual void activemq::commands::ActiveMQQueue::copyDataStructure (const DataStructure * *src*) [virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

6.56.2.5 virtual bool activemq::commands::ActiveMQQueue::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p.1372) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

6.56.2.6 virtual const cms::Destination* activemq::commands::ActiveMQQueue::getCMSDestination () const [inline, virtual]

Returns

the **cms::Destination** (p.1387) interface pointer that the objects that derive from this class implement.

6.56.2.7 virtual const cms::CMSProperties& activemq::commands::ActiveMQQueue::getCMSProperties () const [inline, virtual]

Retrieve any properties that might be part of the destination that was specified.

This is a deviation from the JMS spec but necessary due to C++ restrictions.

Returns

const reference to a properties object.

Implements **cms::Destination** (p.1389).

6.56.2.8 `virtual unsigned char activemq::commands::ActiveMQQueue::getDataStructureType () const`
[virtual]

6.56.2.9 `virtual cms::Destination::DestinationType activemq::commands::ActiveMQQueue::getDestinationType () const`
[inline, virtual]

Retrieve the Destination Type for this Destination.

Returns

The Destination Type

Implements **cms::Destination** (p. 1389).

6.56.2.10 `virtual std::string activemq::commands::ActiveMQQueue::getQueueName () const throw (cms::CMSException)` [inline, virtual]

Gets the name of this queue.

Returns

The queue name.

Implements **cms::Queue** (p. 2511).

6.56.2.11 `virtual std::string activemq::commands::ActiveMQQueue::toString () const` [virtual]

Returns a string containing the information for this **DataStructure** (p. 1372) such as its type and value of its elements.

Returns

formatted string useful for debugging.

6.56.3 Field Documentation

6.56.3.1 `const unsigned char activemq::commands::ActiveMQQueue::ID _ - ACTIVEMQQUEUE = 100` [static]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ActiveMQQueue.h`

6.57 activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 382).


```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQQueueMarshaller.h>
```

Inheritance diagram for activemq:wireformat::openwire::marshal:v3::ActiveMQQueueMarshaller:

Public Member Functions

- **ActiveMQQueueMarshaller** ()
- virtual **~ActiveMQQueueMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.57.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 382). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.57.2 Constructor & Destructor Documentation

6.57.2.1 `activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller::ActiveMQQueueMarshaller () [inline]`

6.57.2.2 `virtual activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller::~~ActiveMQQueueMarshaller () [inline, virtual]`

6.57.3 Member Function Documentation

6.57.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.57.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.57.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller` (p. 251).

6.57.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller` (p. 252).

6.57.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller` (p. 252).

6.57.3.6 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller` (p. 253).

```
6.57.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller` (p. 253).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQQueueMarshaller.h`

6.58 `activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller` Class Reference

Marshaling code for Open Wire Format for `ActiveMQQueueMarshaller` (p. 386).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQQueueMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller`:

Public Member Functions

- **ActiveMQQueueMarshaller** ()
- virtual **~ActiveMQQueueMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.58.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 386). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.58.2 Constructor & Destructor Documentation

6.58.2.1 `activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller::ActiveMQQueueMarshaller () [inline]`

6.58.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller::~~ActiveMQQueueMarshaller () [inline, virtual]`

6.58.3 Member Function Documentation

6.58.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.58.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.58.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller` (p. 255).

6.58.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller` (p. 256).

6.58.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller` (p. 256).

6.58.3.6 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller` (p. 257).

```
6.58.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller` (p. 257).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQQueueMarshaller.h`

6.59 `activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller` Class Reference

Marshaling code for Open Wire Format for `ActiveMQQueueMarshaller` (p. 390).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQQueueMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller`:

Public Member Functions

- **ActiveMQQueueMarshaller** ()
- virtual **~ActiveMQQueueMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.59.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 390). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.59.2 Constructor & Destructor Documentation

6.59.2.1 `activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller::ActiveMQQueueMarshaller () [inline]`

6.59.2.2 `virtual activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller::~~ActiveMQQueueMarshaller () [inline, virtual]`

6.59.3 Member Function Documentation

6.59.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.59.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.59.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller` (p. 259).

6.59.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller` (p. 259).

6.59.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller` (p. 260).

6.59.3.6 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller` (p. 260).

```
6.59.3.7 virtual void ac-
        tivemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller::tightUnmarshal
        ( OpenWireFormat * wireFormat, commands::DataStructure
        * dataStructure, decaf::io::DataInputStream * dataIn,
        utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller` (p. 261).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQQueueMarshaller.h`

6.60 `activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller` Class Reference

Marshaling code for Open Wire Format for `ActiveMQQueueMarshaller` (p. 394).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQQueueMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller`:

Public Member Functions

- **ActiveMQQueueMarshaller** ()
- virtual **~ActiveMQQueueMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.60.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 394). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.60.2 Constructor & Destructor Documentation

6.60.2.1 `activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller::ActiveMQQueueMarshaller () [inline]`

6.60.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller::~~ActiveMQQueueMarshaller () [inline, virtual]`

6.60.3 Member Function Documentation

6.60.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.60.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.60.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller` (p. 263).

6.60.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller` (p. 263).

6.60.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller` (p. 264).

6.60.3.6 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller` (p. 264).

```
6.60.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller` (p. 265).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQQueueMarshaller.h`

6.61 `activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller` Class Reference

Marshaling code for Open Wire Format for `ActiveMQQueueMarshaller` (p. 398).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQQueueMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller`:

Public Member Functions

- **ActiveMQQueueMarshaller** ()
- virtual **~ActiveMQQueueMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.61.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 398). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.61.2 Constructor & Destructor Documentation

6.61.2.1 `activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller::ActiveMQQueueMarshaller () [inline]`

6.61.2.2 `virtual activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller::~~ActiveMQQueueMarshaller () [inline, virtual]`

6.61.3 Member Function Documentation

6.61.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.61.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.61.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller` (p. 267).

6.61.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller` (p. 267).

6.61.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller` (p. 268).

6.61.3.6 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions

- IOException* if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller` (p. 268).

```
6.61.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller` (p. 269).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQQueueMarshaller.h`

6.62 `activemq::core::ActiveMQSession` Class Reference

```
#include <src/main/activemq/core/ActiveMQSession.h>
```

Inheritance diagram for `activemq::core::ActiveMQSession`:

Public Member Functions

- **ActiveMQSession** (const **Pointer**< **commands::SessionInfo** > &sessionInfo, **cms::Session::AcknowledgeMode** ackMode, const **decaf::util::Properties** &properties, **ActiveMQConnection** *connection)
- virtual **~ActiveMQSession** ()
- void **redispatch** (**MessageDispatchChannel** &unconsumedMessages)
Redispatches the given set of unconsumed messages to the consumers.
- void **start** ()
Stops asynchronous message delivery.
- void **stop** ()
Starts asynchronous message delivery.
- bool **isStarted** () const
Indicates whether or not the session is currently in the started state.
- bool **isAutoAcknowledge** () const
- bool **isDupsOkAcknowledge** () const
- bool **isClientAcknowledge** () const
- bool **isIndividualAcknowledge** () const
- void **fire** (const **exceptions::ActiveMQException** &ex)
Fires the given exception to the exception listener of the connection.
- virtual void **dispatch** (const **Pointer**< **MessageDispatch** > &message)
Dispatches a message to a particular consumer.
- virtual void **close** () throw (**cms::CMSEException**)
Closes this session as well as any active child consumers or producers.
- virtual void **commit** () throw (**cms::CMSEException**)
Commits all messages done in this transaction and releases any locks currently held.
- virtual void **rollback** () throw (**cms::CMSEException**)
Rollsback all messages done in this transaction and releases any locks currently held.
- virtual void **recover** () throw (**cms::CMSEException**)
Stops message delivery in this session, and restarts message delivery with the oldest unacknowledged message.
- virtual **cms::MessageConsumer** * **createConsumer** (const **cms::Destination** *destination) throw (**cms::CMSEException**)
Creates a MessageConsumer for the specified destination.
- virtual **cms::MessageConsumer** * **createConsumer** (const **cms::Destination** *destination, const std::string &selector) throw (**cms::CMSEException**)
Creates a MessageConsumer for the specified destination, using a message selector.

- virtual **cms::MessageConsumer** * **createConsumer** (const **cms::Destination** *destination, const std::string &selector, bool noLocal) throw (cms::CMSEException)
Creates a MessageConsumer for the specified destination, using a message selector.
- virtual **cms::MessageConsumer** * **createDurableConsumer** (const **cms::Topic** *destination, const std::string &name, const std::string &selector, bool noLocal=false) throw (cms::CMSEException)
Creates a durable subscriber to the specified topic, using a message selector.
- virtual **cms::MessageProducer** * **createProducer** (const **cms::Destination** *destination) throw (cms::CMSEException)
Creates a MessageProducer to send messages to the specified destination.
- virtual **cms::QueueBrowser** * **createBrowser** (const **cms::Queue** *queue) throw (cms::CMSEException)
Creates a new QueueBrowser to peek at Messages on the given Queue.
- virtual **cms::QueueBrowser** * **createBrowser** (const **cms::Queue** *queue, const std::string &selector) throw (cms::CMSEException)
Creates a new QueueBrowser to peek at Messages on the given Queue.
- virtual **cms::Queue** * **createQueue** (const std::string &queueName) throw (cms::CMSEException)
Creates a queue identity given a Queue name.
- virtual **cms::Topic** * **createTopic** (const std::string &topicName) throw (cms::CMSEException)
Creates a topic identity given a Queue name.
- virtual **cms::TemporaryQueue** * **createTemporaryQueue** () throw (cms::CMSEException)
Creates a TemporaryQueue object.
- virtual **cms::TemporaryTopic** * **createTemporaryTopic** () throw (cms::CMSEException)
Creates a TemporaryTopic object.
- virtual **cms::Message** * **createMessage** () throw (cms::CMSEException)
Creates a new Message.
- virtual **cms::BytesMessage** * **createBytesMessage** () throw (cms::CMSEException)
Creates a BytesMessage.
- virtual **cms::BytesMessage** * **createBytesMessage** (const unsigned char *bytes, std::size_t bytesSize) throw (cms::CMSEException)
Creates a BytesMessage and sets the pay-load to the passed value.
- virtual **cms::StreamMessage** * **createStreamMessage** () throw (cms::CMSEException)
Creates a new StreamMessage.

- virtual **cms::TextMessage** * **createTextMessage** () throw (cms::CMSEException)
Creates a new TextMessage.
- virtual **cms::TextMessage** * **createTextMessage** (const std::string &text) throw (cms::CMSEException)
Creates a new TextMessage and set the text to the value given.
- virtual **cms::MapMessage** * **createMapMessage** () throw (cms::CMSEException)
Creates a new MapMessage.
- virtual **cms::Session::AcknowledgeMode** **getAcknowledgeMode** () const throw (cms::CMSEException)
Returns the acknowledgment mode of the session.
- virtual bool **isTransacted** () const throw (cms::CMSEException)
Gets if the Sessions is a Transacted Session.
- virtual void **unsubscribe** (const std::string &name) throw (cms::CMSEException)
Unsubscribes a durable subscription that has been created by a client.
- void **send** (cms::Message *message, **ActiveMQProducer** *producer, **util::Usage** *usage) throw (cms::CMSEException)
Sends a message from the Producer specified using this session's connection the message will be sent using the best available means depending on the configuration of the connection.
- **cms::ExceptionListener** * **getExceptionListener** ()
This method gets any registered exception listener of this sessions connection and returns it.
- const **commands::SessionInfo** & **getSessionInfo** () const
Gets the Session Information object for this session, if the session is closed than this method throws an exception.
- const **commands::SessionId** & **getSessionId** () const
Gets the Session Id object for this session, if the session is closed than this method throws an exception.
- **ActiveMQConnection** * **getConnection** () const
*Gets the **ActiveMQConnection** (p. 202) that is associated with this session.*
- long long **getLastDeliveredSequenceId** () const
Gets the currently set Last Delivered Sequence Id.
- void **setLastDeliveredSequenceId** (long long value)
Sets the value of the Last Delivered Sequence Id.
- void **oneway** (Pointer< **commands::Command** > command) throw (activemq::exceptions::ActiveMQException)
Sends a oneway message.

- void **syncRequest** (**Pointer**< **commands::Command** > command, unsigned int timeout=0) throw (**activemq::exceptions::ActiveMQException**)
Sends a synchronous request and returns the response from the broker.
- void **disposeOf** (**decaf::lang::Pointer**< **commands::ConsumerId** > id, long long last-DeliveredSequenceId) throw (**activemq::exceptions::ActiveMQException**)
Dispose of a Consumer from this session.
- void **disposeOf** (**decaf::lang::Pointer**< **commands::ProducerId** > id) throw (**activemq::exceptions::ActiveMQException**)
Dispose of a Producer from this session.
- void **doStartTransaction** () throw (**exceptions::ActiveMQException**)
Starts if not already start a Transaction for this Session.
- void **acknowledge** ()
Request that the Session inform all its consumers to Acknowledge all Message's that have been received so far.
- void **deliverAcks** ()
Request that this Session inform all of its consumers to deliver their pending acks.
- void **clearMessagesInProgress** ()
Request that this Session inform all of its consumers to clear all messages that are currently in progress.
- void **wakeup** ()
Causes the Session to wakeup its executor and ensure all messages are dispatched.

Friends

- class **ActiveMQSessionExecutor**

6.62.1 Constructor & Destructor Documentation

- 6.62.1.1** **activemq::core::ActiveMQSession::ActiveMQSession** (const **Pointer**< **commands::SessionInfo** > & *sessionInfo*, **cms::Session::AcknowledgeMode** *ackMode*, const **decaf::util::Properties** & *properties*, **ActiveMQConnection** * *connection*)
- 6.62.1.2** **virtual activemq::core::ActiveMQSession::~~ActiveMQSession** ()
 [virtual]

6.62.2 Member Function Documentation

- 6.62.2.1** void **activemq::core::ActiveMQSession::acknowledge** ()

Request that the Session inform all its consumers to Acknowledge all Message's that have been received so far.

6.62.2.2 void activemq::core::ActiveMQSession::clearMessagesInProgress ()

Request that this Session inform all of its consumers to clear all messages that are currently in progress.

6.62.2.3 virtual void activemq::core::ActiveMQSession::close () throw (cms::CMSException) [virtual]

Closes this session as well as any active child consumers or producers.

Exceptions

CMSException

6.62.2.4 virtual void activemq::core::ActiveMQSession::commit () throw (cms::CMSException) [virtual]

Commits all messages done in this transaction and releases any locks currently held.

Exceptions

CMSException

6.62.2.5 virtual cms::QueueBrowser* activemq::core::ActiveMQSession::createBrowser (const cms::Queue * queue) throw (cms::CMSException) [virtual]

Creates a new QueueBrowser to peek at Messages on the given Queue.

Parameters

queue the Queue to browse

Returns

New QueueBrowser that is owned by the caller.

Exceptions

CMSException - If an internal error occurs.

InvalidDestinationException - if the destination given is invalid.

6.62.2.6 virtual cms::QueueBrowser* activemq::core::ActiveMQSession::createBrowser (const cms::Queue * queue, const std::string & selector) throw (cms::CMSException) [virtual]

Creates a new QueueBrowser to peek at Messages on the given Queue.

Parameters

queue the Queue to browse

selector the Message selector to filter which messages are browsed.

Returns

New QueueBrowser that is owned by the caller.

Exceptions

CMSEException - If an internal error occurs.

InvalidDestinationException - if the destination given is invalid.

```
6.62.2.7 virtual cms::BytesMessage* ac-
        tivemq::core::ActiveMQSession::createBytesMessage (    )
        throw ( cms::CMSEException ) [virtual]
```

Creates a BytesMessage.

Returns

a newly created BytesMessage.

Exceptions

CMSEException

```
6.62.2.8 virtual cms::BytesMessage* ac-
        tivemq::core::ActiveMQSession::createBytesMessage (
        const unsigned char * bytes, std::size_t bytesSize ) throw (
        cms::CMSEException ) [virtual]
```

Creates a BytesMessage and sets the pay-load to the passed value.

Parameters

bytes - an array of bytes to set in the message

bytesSize - the size of the bytes array, or number of bytes to use

Returns

a newly created BytesMessage.

Exceptions

CMSEException

```
6.62.2.9 virtual cms::MessageConsumer* ac-
        tivemq::core::ActiveMQSession::createConsumer ( const
        cms::Destination * destination, const std::string & selector, bool
        noLocal ) throw ( cms::CMSEException ) [virtual]
```

Creates a MessageConsumer for the specified destination, using a message selector.

Parameters

- destination* - The Destination that this consumer receiving messages for.
- selector* - The Message Selector string to use for this destination
- noLocal* - if true, and the destination is a topic, inhibits the delivery of messages published by its own connection. The behavior for NoLocal is not specified if the destination is a queue.

Exceptions

CMSException

6.62.2.10 `virtual cms::MessageConsumer* activemq::core::ActiveMQSession::createConsumer (const cms::Destination * destination, const std::string & selector) throw (cms::CMSException)` [virtual]

Creates a MessageConsumer for the specified destination, using a message selector.

Parameters

- destination* - The Destination that this consumer receiving messages for.
- selector* - The Message Selector string to use for this destination

Exceptions

CMSException

6.62.2.11 `virtual cms::MessageConsumer* activemq::core::ActiveMQSession::createConsumer (const cms::Destination * destination) throw (cms::CMSException)` [virtual]

Creates a MessageConsumer for the specified destination.

Parameters

- destination* - The Destination that this consumer receiving messages for.

Exceptions

CMSException

6.62.2.12 `virtual cms::MessageConsumer* activemq::core::ActiveMQSession::createDurableConsumer (const cms::Topic * destination, const std::string & name, const std::string & selector, bool noLocal = false) throw (cms::CMSException)` [virtual]

Creates a durable subscriber to the specified topic, using a message selector.

Parameters

destination - the topic to subscribe to

name - The name used to identify the subscription

selector - only messages matching the selector are received

noLocal - if true, and the destination is a topic, inhibits the delivery of messages published by its own connection. The behavior for NoLocal is not specified if the destination is a queue.

Exceptions

CMSEException

6.62.2.13 `virtual cms::MapMessage* activemq::core::ActiveMQSession::createMapMessage ()
throw (cms::CMSEException) [virtual]`

Creates a new MapMessage.

Returns

a newly created MapMessage.

Exceptions

CMSEException

6.62.2.14 `virtual cms::Message* activemq::core::ActiveMQSession::createMessage ()
throw (cms::CMSEException) [virtual]`

Creates a new Message.

Exceptions

CMSEException

6.62.2.15 `virtual cms::MessageProducer* activemq::core::ActiveMQSession::createProducer (const
cms::Destination * destination) throw (cms::CMSEException)
[virtual]`

Creates a MessageProducer to send messages to the specified destination.

Parameters

destination - the Destination to publish on

Exceptions

CMSEException

6.62.2.16 virtual cms::Queue* activemq::core::ActiveMQSession::createQueue
(const std::string & *queueName*) throw (cms::CMSException)
[virtual]

Creates a queue identity given a Queue name.

Parameters

queueName - the name of the new Queue

Exceptions

CMSException

6.62.2.17 virtual cms::StreamMessage* activemq::core::ActiveMQSession::createStreamMessage () throw (cms::CMSException) [virtual]

Creates a new StreamMessage.

Returns

a newly created StreamMessage.

Exceptions

CMSException

6.62.2.18 virtual cms::TemporaryQueue* activemq::core::ActiveMQSession::createTemporaryQueue () throw (cms::CMSException) [virtual]

Creates a TemporaryQueue object.

Exceptions

CMSException

6.62.2.19 virtual cms::TemporaryTopic* activemq::core::ActiveMQSession::createTemporaryTopic () throw (cms::CMSException) [virtual]

Creates a TemporaryTopic object.

Exceptions

CMSException

6.62.2.20 `virtual cms::TextMessage* activemq::core::ActiveMQSession::createTextMessage ()
throw (cms::CMSException) [virtual]`

Creates a new TextMessage.

Returns

a newly created TextMessage.

Exceptions

CMSException

6.62.2.21 `virtual cms::TextMessage* activemq::core::ActiveMQSession::createTextMessage (const std::string & text) throw (cms::CMSException) [virtual]`

Creates a new TextMessage and set the text to the value given.

Parameters

text - The initial text for the message

Returns

a newly created TextMessage with the given Text set in the Message body.

Exceptions

CMSException

6.62.2.22 `virtual cms::Topic* activemq::core::ActiveMQSession::createTopic (const std::string & topicName) throw (cms::CMSException) [virtual]`

Creates a topic identity given a Queue name.

Parameters

topicName - the name of the new Topic

Exceptions

CMSException

6.62.2.23 `void activemq::core::ActiveMQSession::deliverAcks ()`

Request that this Session inform all of its consumers to deliver their pending acks.

6.62.2.24 `virtual void activemq::core::ActiveMQSession::dispatch (const Pointer< MessageDispatch > & message) [virtual]`

Dispatches a message to a particular consumer.

Parameters

message - the message to be dispatched

Implements `activemq::core::Dispatcher` (p. 1441).

6.62.2.25 `void activemq::core::ActiveMQSession::disposeOf (decaf::lang::Pointer< commands::ConsumerId > id, long long lastDeliveredSequenceId) throw (activemq::exceptions::ActiveMQException)`

Dispose of a Consumer from this session.

Removes it from the Connection and clean up any resources associated with it.

Parameters

id The Id of the Consumer to dispose.

lastDeliveredSequenceId The Broker Sequence Id of the last message the Consumer delivered.

Exceptions

ActiveMQException if an internal error occurs.

6.62.2.26 `void activemq::core::ActiveMQSession::disposeOf (decaf::lang::Pointer< commands::ProducerId > id) throw (activemq::exceptions::ActiveMQException)`

Dispose of a Producer from this session.

Removes it from the Connection and clean up any resources associated with it.

Parameters

id - the Id of the Producer to dispose.

Exceptions

ActiveMQException if an internal error occurs.

6.62.2.27 `void activemq::core::ActiveMQSession::doStartTransaction () throw (exceptions::ActiveMQException)`

Starts if not already start a Transaction for this Session.

If the session is not a Transacted Session then an exception is thrown. If a transaction is already in progress then this method has no effect.

Exceptions

ActiveMQException if this is not a Transacted Session.

6.62.2.28 `void activemq::core::ActiveMQSession::fire (const exceptions::ActiveMQException & ex)`

Fires the given exception to the exception listener of the connection.

6.62.2.29 `virtual cms::Session::AcknowledgeMode activemq::core::ActiveMQSession::getAcknowledgeMode () const throw (cms::CMSException) [virtual]`

Returns the acknowledgment mode of the session.

Returns

the Sessions Acknowledge Mode

6.62.2.30 `ActiveMQConnection* activemq::core::ActiveMQSession::getConnection () const [inline]`

Gets the **ActiveMQConnection** (p. 202) that is associated with this session.

6.62.2.31 `cms::ExceptionListener* activemq::core::ActiveMQSession::getExceptionListener ()`

This method gets any registered exception listener of this sessions connection and returns it.

Mainly intended for use by the objects that this session creates so that they can notify the client of exceptions that occur in the context of another thread.

Returns

cms::ExceptionListener (p. 1483) pointer or NULL

6.62.2.32 `long long activemq::core::ActiveMQSession::getLastDeliveredSequenceId () const [inline]`

Gets the currently set Last Delivered Sequence Id.

Returns

long long containing the sequence id of the last delivered Message.

6.62.2.33 `const commands::SessionId& activemq::core::ActiveMQSession::getSessionId () const [inline]`

Gets the Session Id object for this session, if the session is closed than this method throws an exception.

Returns

SessionId Reference

6.62.2.34 `const commands::SessionInfo& activemq::core::ActiveMQSession::getSessionInfo () const`
[inline]

Gets the Session Information object for this session, if the session is closed than this method throws an exception.

Returns

SessionInfo Reference

6.62.2.35 `bool activemq::core::ActiveMQSession::isAutoAcknowledge () const`
[inline]

6.62.2.36 `bool activemq::core::ActiveMQSession::isClientAcknowledge () const`
[inline]

6.62.2.37 `bool activemq::core::ActiveMQSession::isDupsOkAcknowledge () const`
[inline]

6.62.2.38 `bool activemq::core::ActiveMQSession::isIndividualAcknowledge () const` [inline]

6.62.2.39 `bool activemq::core::ActiveMQSession::isStarted () const`

Indicates whether or not the session is currently in the started state.

6.62.2.40 `virtual bool activemq::core::ActiveMQSession::isTransacted () const`
`throw (cms::CMSException)` [virtual]

Gets if the Sessions is a Transacted Session.

Returns

transacted true - false.

6.62.2.41 `void activemq::core::ActiveMQSession::oneway (`
`Pointer< commands::Command > command) throw (`
`activemq::exceptions::ActiveMQException)`

Sends a oneway message.

Parameters

command The message to send.

Exceptions

ActiveMQException if not currently connected, or if the operation fails for any reason.

6.62.2.42 `virtual void activemq::core::ActiveMQSession::recover () throw (cms::CMSException) [virtual]`

Stops message delivery in this session, and restarts message delivery with the oldest unacknowledged message.

All consumers deliver messages in a serial order. Acknowledging a received message automatically acknowledges all messages that have been delivered to the client.

Restarting a session causes it to take the following actions:

- Stop message delivery
- Mark all messages that might have been delivered but not acknowledged as "redelivered"
- Restart the delivery sequence including all unacknowledged messages that had been previously delivered. Redelivered messages do not have to be delivered in exactly their original delivery order.

Exceptions

CMSException - if the CMS provider fails to stop and restart message delivery due to some internal error.

IllegalStateException - if the method is called by a transacted session.

6.62.2.43 `void activemq::core::ActiveMQSession::redispatch (MessageDispatchChannel & unconsumedMessages)`

Redispatches the given set of unconsumed messages to the consumers.

Parameters

unconsumedMessages - unconsumed messages to be redelivered.

6.62.2.44 `virtual void activemq::core::ActiveMQSession::rollback () throw (cms::CMSException) [virtual]`

Rollsback all messages done in this transaction and releases any locks currently held.

Exceptions

CMSException

6.62.2.45 `void activemq::core::ActiveMQSession::send (cms::Message * message, ActiveMQProducer * producer, util::Usage * usage) throw (cms::CMSException)`

Sends a message from the Producer specified using this session's connection the message will be sent using the best available means depending on the configuration of the connection.

Asynchronous sends will be chosen if at all possible.

Parameters

message The message to send to the broker.

producer The sending Producer

usage Pointer to a Usage tracker which if set will be increased by the size of the given message.

Exceptions

CMSException

6.62.2.46 `void activemq::core::ActiveMQSession::setLastDeliveredSequenceId (long long value) [inline]`

Sets the value of the Last Delivered Sequence Id.

Parameters

value The new value to assign to the Last Delivered Sequence Id property.

6.62.2.47 `void activemq::core::ActiveMQSession::start ()`

Stops asynchronous message delivery.

6.62.2.48 `void activemq::core::ActiveMQSession::stop ()`

Starts asynchronous message delivery.

6.62.2.49 `void activemq::core::ActiveMQSession::syncRequest (Pointer< commands::Command > command, unsigned int timeout = 0) throw (activemq::exceptions::ActiveMQException)`

Sends a synchronous request and returns the response from the broker.

Converts any error responses into an exception.

Parameters

command The request command.

timeout The time to wait for a response, default is zero or infinite.

Exceptions

ActiveMQException thrown if an error response was received from the broker, or if any other error occurred.

6.62.2.50 `virtual void activemq::core::ActiveMQSession::unsubscribe (const std::string & name) throw (cms::CMSException) [virtual]`

Unsubscribes a durable subscription that has been created by a client.

This method deletes the state being maintained on behalf of the subscriber by its provider. It is erroneous for a client to delete a durable subscription while there is an active MessageConsumer or Subscriber for the subscription, or while a consumed message is part of a pending transaction or has not been acknowledged in the session.

Parameters

name the name used to identify this subscription

Exceptions

CMSException

6.62.2.51 `void activemq::core::ActiveMQSession::wakeup ()`

Causes the Session to wakeup its executor and ensure all messages are dispatched.

6.62.3 Friends And Related Function Documentation

6.62.3.1 `friend class ActiveMQSessionExecutor [friend]`

The documentation for this class was generated from the following file:

- `src/main/activemq/core/ActiveMQSession.h`

6.63 activemq::core::ActiveMQSessionExecutor Class Reference

Delegate dispatcher for a single session.

```
#include <src/main/activemq/core/ActiveMQSessionExecutor.h>
```

Inheritance diagram for `activemq::core::ActiveMQSessionExecutor`:

Public Member Functions

- `ActiveMQSessionExecutor (ActiveMQSession *session)`
Creates an un-started executor for the given session.
- `virtual ~ActiveMQSessionExecutor ()`
Calls `stop()` (p. 421) then `clear()` (p. 420).
- `virtual void execute (const Pointer< MessageDispatch > &data)`
Executes the dispatch.

- virtual void **executeFirst** (const **Pointer**< **MessageDispatch** > &data)
Executes the dispatch.
- virtual void **clearMessagesInProgress** ()
Removes all messages in the Dispatch Channel so that non are delivered.
- virtual bool **hasUnconsumedMessages** () const
- virtual void **wakeup** ()
wakeup this executor and dispatch any pending messages.
- virtual void **start** ()
Starts the dispatching.
- virtual void **stop** ()
Stops dispatching.
- virtual void **close** ()
Terminates the dispatching thread.
- virtual bool **isRunning** () const
- virtual bool **isEmpty** ()
- virtual void **clear** ()
Removes all queued messages and destroys them.
- virtual bool **iterate** ()
*Iterates on the **MessageDispatchChannel** (p. 2088) sending all pending messages to the Consumers they are destined for.*
- std::vector< **Pointer**< **MessageDispatch** > > **getUnconsumedMessages** ()

6.63.1 Detailed Description

Delegate dispatcher for a single session. Contains a thread to provide for asynchronous dispatching.

6.63.2 Constructor & Destructor Documentation

6.63.2.1 **activemq::core::ActiveMQSessionExecutor::ActiveMQSessionExecutor** (**ActiveMQSession** * *session*)

Creates an un-started executor for the given session.

6.63.2.2 **virtual** **activemq::core::ActiveMQSessionExecutor::~ActiveMQSessionExecutor** () [virtual]

Calls **stop()** (p. 421) then **clear()** (p. 420).

6.63.3 Member Function Documentation

6.63.3.1 `virtual void activemq::core::ActiveMQSessionExecutor::clear ()`
[inline, virtual]

Removes all queued messages and destroys them.

6.63.3.2 `virtual void activemq::core::ActiveMQSessionExecutor::clearMessagesInProgress ()`
[inline, virtual]

Removes all messages in the Dispatch Channel so that non are delivered.

6.63.3.3 `virtual void activemq::core::ActiveMQSessionExecutor::close ()`
[inline, virtual]

Terminates the dispatching thread.

Once this is called, the executor is no longer usable.

6.63.3.4 `virtual void activemq::core::ActiveMQSessionExecutor::execute (const Pointer< MessageDispatch > & data)` [virtual]

Executes the dispatch.

Adds the given data to the end of the queue.

Parameters

data - the data to be dispatched.

6.63.3.5 `virtual void activemq::core::ActiveMQSessionExecutor::executeFirst (const Pointer< MessageDispatch > & data)` [virtual]

Executes the dispatch.

Adds the given data to the beginning of the queue.

Parameters

data - the data to be dispatched.

6.63.3.6 `std::vector< Pointer<MessageDispatch> > activemq::core::ActiveMQSessionExecutor::getUnconsumedMessages ()`
[inline]

Returns

a vector containing all the unconsumed messages, this clears the Message Dispatch Channel when called.

6.63.3.7 `virtual bool activemq::core::ActiveMQSessionExecutor::hasUnconsumedMessages ()`
`const [inline, virtual]`

Returns

true if there are any pending messages in the dispatch channel.

6.63.3.8 `virtual bool activemq::core::ActiveMQSessionExecutor::isEmpty ()`
`[inline, virtual]`

Returns

true if there are no messages in the Dispatch Channel.

6.63.3.9 `virtual bool activemq::core::ActiveMQSessionExecutor::isRunning ()`
`const [inline, virtual]`

Returns

true indicates if the executor is started

6.63.3.10 `virtual bool activemq::core::ActiveMQSessionExecutor::iterate ()`
`[virtual]`

Iterates on the **MessageDispatchChannel** (p. 2088) sending all pending messages to the Consumers they are destined for.

Returns

false if there are no more messages to dispatch.

Implements **activemq::threads::Task** (p. 2956).

6.63.3.11 `virtual void activemq::core::ActiveMQSessionExecutor::start ()`
`[virtual]`

Starts the dispatching.

6.63.3.12 `virtual void activemq::core::ActiveMQSessionExecutor::stop ()`
`[virtual]`

Stops dispatching.

6.63.3.13 `virtual void activemq::core::ActiveMQSessionExecutor::wakeup ()`
`[virtual]`

wakeup this executer and dispatch any pending messages.

The documentation for this class was generated from the following file:

- `src/main/activemq/core/ActiveMQSessionExecutor.h`

6.64 activemq::commands::ActiveMQStreamMessage Class Reference

```
#include <src/main/activemq/commands/ActiveMQStreamMessage.h>
```

Inheritance diagram for activemq::commands::ActiveMQStreamMessage:

Public Member Functions

- **ActiveMQStreamMessage** ()
- virtual **~ActiveMQStreamMessage** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **ActiveMQStreamMessage * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1372) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent.*
- virtual void **onSend** ()
Store the Data that was written to the stream before a send.
- virtual **cms::StreamMessage * clone** () const
Clone this message exactly, returns a new instance that the caller is required to delete.
- virtual void **clearBody** () throw (cms::CMSException)
Clears out the body of the message.
- virtual void **reset** () throw (cms::CMSException)
Puts the message body in read-only mode and repositions the stream of bytes to the beginning.
- virtual bool **readBoolean** () const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException)
Reads a Boolean from the Stream message stream.
- virtual void **writeBoolean** (bool value) throw (cms::MessageNotWriteableException, cms::CMSException)
Writes a boolean to the Stream message stream as a 1-byte value.

- virtual unsigned char **readByte** () const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException)
Reads a Byte from the Stream message stream.
- virtual void **writeByte** (unsigned char value) throw (cms::MessageNotWriteableException, cms::CMSEException)
Writes a byte to the Stream message stream as a 1-byte value.
- virtual std::size_t **readBytes** (std::vector< unsigned char > &value) const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException)
Reads a byte array from the Stream message stream.
- virtual void **writeBytes** (const std::vector< unsigned char > &value) throw (cms::MessageNotWriteableException, cms::CMSEException)
Writes a byte array to the Stream message stream using the vector size as the number of bytes to write.
- virtual std::size_t **readBytes** (unsigned char *buffer, std::size_t length) const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException)
Reads a portion of the Stream message stream.
- virtual void **writeBytes** (const unsigned char *value, std::size_t offset, std::size_t length) throw (cms::MessageNotWriteableException, cms::CMSEException)
Writes a portion of a byte array to the Stream message stream.
- virtual char **readChar** () const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException)
Reads a Char from the Stream message stream.
- virtual void **writeChar** (char value) throw (cms::MessageNotWriteableException, cms::CMSEException)
Writes a char to the Stream message stream as a 1-byte value.
- virtual float **readFloat** () const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException)
Reads a 32 bit float from the Stream message stream.
- virtual void **writeFloat** (float value) throw (cms::MessageNotWriteableException, cms::CMSEException)
Writes a float to the Stream message stream as a 4 byte value.
- virtual double **readDouble** () const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException)
Reads a 64 bit double from the Stream message stream.

- virtual void **writeDouble** (double value) throw (cms::MessageNotWriteableException, cms::CMSEException)
Writes a double to the Stream message stream as a 8 byte value.
- virtual short **readShort** () const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException)
Reads a 16 bit signed short from the Stream message stream.
- virtual void **writeShort** (short value) throw (cms::MessageNotWriteableException, cms::CMSEException)
Writes a signed short to the Stream message stream as a 2 byte value.
- virtual unsigned short **readUnsignedShort** () const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException)
Reads a 16 bit unsigned short from the Stream message stream.
- virtual void **writeUnsignedShort** (unsigned short value) throw (cms::MessageNotWriteableException, cms::CMSEException)
Writes a unsigned short to the Stream message stream as a 2 byte value.
- virtual int **readInt** () const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException)
Reads a 32 bit signed integer from the Stream message stream.
- virtual void **writeInt** (int value) throw (cms::MessageNotWriteableException, cms::CMSEException)
Writes a signed int to the Stream message stream as a 4 byte value.
- virtual long long **readLong** () const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException)
Reads a 64 bit long from the Stream message stream.
- virtual void **writeLong** (long long value) throw (cms::MessageNotWriteableException, cms::CMSEException)
Writes a long long to the Stream message stream as a 8 byte value.
- virtual std::string **readString** () const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException)
Reads an ASCII String from the Stream message stream.
- virtual void **writeString** (const std::string &value) throw (cms::MessageNotWriteableException, cms::CMSEException)
Writes an ASCII String to the Stream message stream.

Static Public Attributes

- static const unsigned char `ID_ACTIVEMQSTREAMMESSAGE` = 27

6.64.1 Constructor & Destructor Documentation

6.64.1.1 `activemq::commands::ActiveMQStreamMessage::ActiveMQStreamMessage ()`

6.64.1.2 `virtual activemq::commands::ActiveMQStreamMessage::~~ActiveMQStreamMessage () [virtual]`

6.64.2 Member Function Documentation

6.64.2.1 `virtual void activemq::commands::ActiveMQStreamMessage::clearBody () throw (cms::CMSException) [virtual]`

Clears out the body of the message.

This does not clear the headers or properties.

Reimplemented from `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 331).

6.64.2.2 `virtual cms::StreamMessage* activemq::commands::ActiveMQStreamMessage::clone () const [inline, virtual]`

Clone this message exactly, returns a new instance that the caller is required to delete.

Returns

new copy of this message

6.64.2.3 `virtual ActiveMQStreamMessage* activemq::commands::ActiveMQStreamMessage::cloneDataStructure () const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Reimplemented from `activemq::commands::Message` (p. 2023).

6.64.2.4 `virtual void activemq::commands::ActiveMQStreamMessage::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Reimplemented from `activemq::commands::Message` (p. 2023).

6.64.2.5 `virtual bool activemq::commands::ActiveMQStreamMessage::equals (const DataStructure * value) const` [virtual]

Compares the `DataStructure` (p. 1372) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if `DataStructure`'s are Equal.

Reimplemented from `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 331).

6.64.2.6 `virtual unsigned char activemq::commands::ActiveMQStreamMessage::getDataStructureType () const` [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns

new `DataStructure` (p. 1372) type copy.

Reimplemented from `activemq::commands::Message` (p. 2025).

6.64.2.7 `virtual void activemq::commands::ActiveMQStreamMessage::onSend ()` [virtual]

Store the Data that was written to the stream before a send.

Reimplemented from `activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 338).

6.64.2.8 `virtual bool activemq::commands::ActiveMQStreamMessage::readBoolean () const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException)` [virtual]

Reads a Boolean from the Stream message stream.

Returns

boolean value from stream

Exceptions

CMSEException - if the CMS provider fails to read the message due to some internal error.

MessageEOFException - if unexpected end of message stream has been reached.

MessageFormatException - if this type conversion is invalid.

MessageNotReadableException - if the message is in write-only mode.

6.64.2.9 virtual unsigned char activemq::commands::ActiveMQStreamMessage::readByte () const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException) [virtual]

Reads a Byte from the Stream message stream.

Returns

unsigned char value from stream

Exceptions

CMSException - if the CMS provider fails to read the message due to some internal error.

MessageEOFException - if unexpected end of message stream has been reached.

MessageFormatException - if this type conversion is invalid.

MessageNotReadableException - if the message is in write-only mode.

6.64.2.10 virtual std::size_t activemq::commands::ActiveMQStreamMessage::readBytes (unsigned char * *buffer*, std::size_t *length*) const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException) [virtual]

Reads a portion of the Stream message stream.

If the length of array value is less than the number of bytes remaining to be read from the stream, the array should be filled. A subsequent call reads the next increment, and so on.

If the number of bytes remaining in the stream is less than the length of array value, the bytes should be read into the array. The return value of the total number of bytes read will be less than the length of the array, indicating that there are no more bytes left to be read from the stream. The next read of the stream returns -1.

If length is negative, or length is greater than the length of the array value, then an CMSException is thrown. No bytes will be read from the stream for this exception case.

Parameters

buffer the buffer into which the data is read

length the number of bytes to read; must be less than or equal to value.length

Returns

the total number of bytes read into the buffer, or -1 if there is no more data because the end of the stream has been reached

Exceptions

CMSException - if the CMS provider fails to read the message due to some internal error.

MessageEOFException - if unexpected end of message stream has been reached.

MessageFormatException - if this type conversion is invalid.

MessageNotReadableException - if the message is in write-only mode.

6.64.2.11 `virtual std::size_t activemq::commands::ActiveMQStreamMessage::readBytes (std::vector< unsigned char > & value) const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException) [virtual]`

Reads a byte array from the Stream message stream.

If the length of vector value is less than the number of bytes remaining to be read from the stream, the vector should be filled. A subsequent call reads the next increment, and so on.

If the number of bytes remaining in the stream is less than the length of vector value, the bytes should be read into the vector. The return value of the total number of bytes read will be less than the length of the vector, indicating that there are no more bytes left to be read from the stream. The next read of the stream returns -1.

Parameters

value buffer to place data in

Returns

the total number of bytes read into the buffer, or -1 if there is no more data because the end of the stream has been reached

Exceptions

CMSException - if the CMS provider fails to read the message due to some internal error.

MessageEOFException - if unexpected end of message stream has been reached.

MessageFormatException - if this type conversion is invalid.

MessageNotReadableException - if the message is in write-only mode.

6.64.2.12 `virtual char activemq::commands::ActiveMQStreamMessage::readChar () const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException) [virtual]`

Reads a Char from the Stream message stream.

Returns

char value from stream

Exceptions

CMSException - if the CMS provider fails to read the message due to some internal error.

MessageEOFException - if unexpected end of message stream has been reached.

MessageFormatException - if this type conversion is invalid.

MessageNotReadableException - if the message is in write-only mode.

6.64.2.13 virtual double activemq::commands::ActiveMQStreamMessage::readDouble
() const throw (cms::MessageEOFException,
cms::MessageFormatException, cms::MessageNotReadableException,
cms::CMSException) [virtual]

Reads a 64 bit double from the Stream message stream.

Returns

double value from stream

Exceptions

CMSException - if the CMS provider fails to read the message due to some internal error.

MessageEOFException - if unexpected end of message stream has been reached.

MessageFormatException - if this type conversion is invalid.

MessageNotReadableException - if the message is in write-only mode.

6.64.2.14 virtual float activemq::commands::ActiveMQStreamMessage::readFloat
() const throw (cms::MessageEOFException,
cms::MessageFormatException, cms::MessageNotReadableException,
cms::CMSException) [virtual]

Reads a 32 bit float from the Stream message stream.

Returns

double value from stream

Exceptions

CMSException - if the CMS provider fails to read the message due to some internal error.

MessageEOFException - if unexpected end of message stream has been reached.

MessageFormatException - if this type conversion is invalid.

MessageNotReadableException - if the message is in write-only mode.

6.64.2.15 virtual int activemq::commands::ActiveMQStreamMessage::readInt
() const throw (cms::MessageEOFException,
cms::MessageFormatException, cms::MessageNotReadableException,
cms::CMSException) [virtual]

Reads a 32 bit signed integer from the Stream message stream.

Returns

int value from stream

Exceptions

CMSException - if the CMS provider fails to read the message due to some internal error.

MessageEOFException - if unexpected end of message stream has been reached.

MessageFormatException - if this type conversion is invalid.

MessageNotReadableException - if the message is in write-only mode.

6.64.2.16 virtual long long activemq::commands::ActiveMQStreamMessage::readLong
 () const throw (cms::MessageEOFException,
 cms::MessageFormatException, cms::MessageNotReadableException,
 cms::CMSException) [virtual]

Reads a 64 bit long from the Stream message stream.

Returns

long long value from stream

Exceptions

CMSException - if the CMS provider fails to read the message due to some internal error.

MessageEOFException - if unexpected end of message stream has been reached.

MessageFormatException - if this type conversion is invalid.

MessageNotReadableException - if the message is in write-only mode.

6.64.2.17 virtual short activemq::commands::ActiveMQStreamMessage::readShort
 () const throw (cms::MessageEOFException,
 cms::MessageFormatException, cms::MessageNotReadableException,
 cms::CMSException) [virtual]

Reads a 16 bit signed short from the Stream message stream.

Returns

short value from stream

Exceptions

CMSException - if the CMS provider fails to read the message due to some internal error.

MessageEOFException - if unexpected end of message stream has been reached.

MessageFormatException - if this type conversion is invalid.

MessageNotReadableException - if the message is in write-only mode.

6.64.2.18 virtual std::string activemq::commands::ActiveMQStreamMessage::readString () const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException) [virtual]

Reads an ASCII String from the Stream message stream.

Returns

String from stream

Exceptions

CMSException - if the CMS provider fails to read the message due to some internal error.

MessageEOFException - if unexpected end of message stream has been reached.

MessageFormatException - if this type conversion is invalid.

MessageNotReadableException - if the message is in write-only mode.

6.64.2.19 virtual unsigned short activemq::commands::ActiveMQStreamMessage::readUnsignedShort () const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException) [virtual]

Reads a 16 bit unsigned short from the Stream message stream.

Returns

unsigned short value from stream

Exceptions

CMSException - if the CMS provider fails to read the message due to some internal error.

MessageEOFException - if unexpected end of message stream has been reached.

MessageFormatException - if this type conversion is invalid.

MessageNotReadableException - if the message is in write-only mode.

6.64.2.20 virtual void activemq::commands::ActiveMQStreamMessage::reset () throw (cms::CMSException) [virtual]

Puts the message body in read-only mode and repositions the stream of bytes to the beginning.

Exceptions

CMSException

6.64.2.21 `virtual std::string activemq::commands::ActiveMQStreamMessage::toString () const [virtual]`

Returns a string containing the information for this **DataStructure** (p.1372) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::Message** (p. 2033).

6.64.2.22 `virtual void activemq::commands::ActiveMQStreamMessage::writeBoolean (bool value) throw (cms::MessageNotWriteableException, cms::CMSException) [virtual]`

Writes a boolean to the Stream message stream as a 1-byte value.

The value true is written as the value (byte)1; the value false is written as the value (byte)0.

Parameters

value boolean to write to the stream

Exceptions

CMSException - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException - if the message is in read-only mode.

6.64.2.23 `virtual void activemq::commands::ActiveMQStreamMessage::writeByte (unsigned char value) throw (cms::MessageNotWriteableException, cms::CMSException) [virtual]`

Writes a byte to the Stream message stream as a 1-byte value.

Parameters

value byte to write to the stream

Exceptions

CMSException - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException - if the message is in read-only mode.

6.64.2.24 `virtual void activemq::commands::ActiveMQStreamMessage::writeBytes (const unsigned char * value, std::size_t offset, std::size_t length) throw (cms::MessageNotWriteableException, cms::CMSException) [virtual]`

Writes a portion of a byte array to the Stream message stream.

size as the number of bytes to write.

Parameters

value bytes to write to the stream
offset the initial offset within the byte array
length the number of bytes to use

Exceptions

CMSException - if the CMS provider fails to write the message due to some internal error.
MessageNotWriteableException - if the message is in read-only mode.

6.64.2.25 virtual void activemq::commands::ActiveMQStreamMessage::writeBytes
(const std::vector< unsigned char > & *value*) throw (cms::MessageNotWriteableException, cms::CMSException) [virtual]

Writes a byte array to the Stream message stream using the vector size as the number of bytes to write.

Parameters

value bytes to write to the stream

Exceptions

CMSException - if the CMS provider fails to write the message due to some internal error.
MessageNotWriteableException - if the message is in read-only mode.

6.64.2.26 virtual void activemq::commands::ActiveMQStreamMessage::writeChar
(char *value*) throw (cms::MessageNotWriteableException, cms::CMSException) [virtual]

Writes a char to the Stream message stream as a 1-byte value.

Parameters

value char to write to the stream

Exceptions

CMSException - if the CMS provider fails to write the message due to some internal error.
MessageNotWriteableException - if the message is in read-only mode.

6.64.2.27 virtual void activemq::commands::ActiveMQStreamMessage::writeDouble
(double *value*) throw (cms::MessageNotWriteableException, cms::CMSException) [virtual]

Writes a double to the Stream message stream as a 8 byte value.

Parameters

value double to write to the stream

Exceptions

CMSException - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException - if the message is in read-only mode.

6.64.2.28 virtual void activemq::commands::ActiveMQStreamMessage::writeFloat
(float *value*) throw (cms::MessageNotWriteableException,
cms::CMSException) [virtual]

Writes a float to the Stream message stream as a 4 byte value.

Parameters

value float to write to the stream

Exceptions

CMSException - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException - if the message is in read-only mode.

6.64.2.29 virtual void activemq::commands::ActiveMQStreamMessage::writeInt
(int *value*) throw (cms::MessageNotWriteableException,
cms::CMSException) [virtual]

Writes a signed int to the Stream message stream as a 4 byte value.

Parameters

value signed int to write to the stream

Exceptions

CMSException - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException - if the message is in read-only mode.

6.64.2.30 virtual void activemq::commands::ActiveMQStreamMessage::writeLong
(long long *value*) throw (cms::MessageNotWriteableException,
cms::CMSException) [virtual]

Writes a long long to the Stream message stream as a 8 byte value.

Parameters

value signed long long to write to the stream

Exceptions

CMSException - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException - if the message is in read-only mode.

6.64.2.31 virtual void activemq::commands::ActiveMQStreamMessage::writeShort (short *value*) throw (cms::MessageNotWriteableException, cms::CMSException) [virtual]

Writes a signed short to the Stream message stream as a 2 byte value.

Parameters

value signed short to write to the stream

Exceptions

CMSException - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException - if the message is in read-only mode.

6.64.2.32 virtual void activemq::commands::ActiveMQStreamMessage::writeString (const std::string & *value*) throw (cms::MessageNotWriteableException, cms::CMSException) [virtual]

Writes an ASCII String to the Stream message stream.

Parameters

value String to write to the stream

Exceptions

CMSException - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException - if the message is in read-only mode.

6.64.2.33 virtual void activemq::commands::ActiveMQStreamMessage::writeUnsignedShort (unsigned short *value*) throw (cms::MessageNotWriteableException, cms::CMSException) [virtual]

Writes a unsigned short to the Stream message stream as a 2 byte value.

Parameters

value unsigned short to write to the stream

Exceptions

CMSException - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException - if the message is in read-only mode.

6.64.3 Field Documentation

6.64.3.1 const unsigned char activemq::commands::ActiveMQStreamMessage::ID _ - ACTIVEMQSTREAMMESSAGE = 27 [static]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ActiveMQStreamMessage.h`

6.65 `activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller` Class Reference

Marshaling code for Open Wire Format for `ActiveMQStreamMessageMarshaller` (p. 436).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQStreamMessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller`:

Public Member Functions

- `ActiveMQStreamMessageMarshaller ()`
- `virtual ~ActiveMQStreamMessageMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaller.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write a object instance to data output stream.
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)`
Write a object instance to data output stream.

6.65.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 436).
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.65.2 Constructor & Destructor Documentation

6.65.2.1 `activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller::ActiveMQStreamMessageMarshaller()` [inline]

6.65.2.2 `virtual activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller::~ActiveMQStreamMessageMarshaller()` [inline, virtual]

6.65.3 Member Function Documentation

6.65.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1331).

6.65.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1336).

6.65.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut)` throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2176).

6.65.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2176).

6.65.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2177).

6.66

activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller

Class Reference

441

```
6.65.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller::tightMars
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2178).

```
6.65.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller::tightUnm
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 2178).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQStreamMessageMarshaller.h

6.66 activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMes Class Reference

Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 439).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQStreamMessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller`:

Public Member Functions

- **ActiveMQStreamMessageMarshaller** ()
- virtual **~ActiveMQStreamMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.66.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 439).
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module

6.66.2 Constructor & Destructor Documentation

6.66.2.1 `activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller::ActiveMQStreamMessageMarshaller () [inline]`

6.66.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller::~~ActiveMQStreamMessageMarshaller () [inline, virtual]`

6.66.3 Member Function Documentation

6.66.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.66.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.66.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller::marshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2184).

6.66.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2184).

6.66.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2185).

6.66.3.6 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

6.67

activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller

Class Reference

445

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions

- IOException* if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2186).

6.66.3.7 virtual void ac-

```
ativemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller::tightUnm  
( OpenWireFormat * wireFormat, commands::DataStructure  
* dataStructure, decaf::io::DataInputStream * dataIn,  
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 2186).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**ActiveMQStreamMessageMarshaller.h**

6.67 activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMes Class Reference

Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 443).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQStreamMessageMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller**:

Public Member Functions

- **ActiveMQStreamMessageMarshaller** ()
- virtual **~ActiveMQStreamMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.67.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 443).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.67.2 Constructor & Destructor Documentation

6.67.2.1 `activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller::ActiveMQStreamMessageMarshaller () [inline]`

6.67.2.2 `virtual activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller::~ActiveMQStreamMessageMarshaller () [inline, virtual]`

6.67.3 Member Function Documentation

6.67.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.67.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.67.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller::marshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2172).

6.67.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2172).

6.67.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2173).

6.67.3.6 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

6.68

activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller

Class Reference

449

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions

- IOException* if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2174).

6.67.3.7 virtual void ac-

```
activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller::tightUnm  
( OpenWireFormat * wireFormat, commands::DataStructure  
* dataStructure, decaf::io::DataInputStream * dataIn,  
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::MessageMarshaller** (p. 2174).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQStreamMessageMarshaller.h

6.68 activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMes Class Reference

Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 447).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQStreamMessageMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller**:

Public Member Functions

- **ActiveMQStreamMessageMarshaller** ()
- virtual **~ActiveMQStreamMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const

Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Un-marshall an object instance from the data input stream.

- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

Un-marshall an object instance from the data input stream.

- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.68.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 447).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.68.2 Constructor & Destructor Documentation

6.68.2.1 `activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller::ActiveMQStreamMessageMarshaller () [inline]`

6.68.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller::~ActiveMQStreamMessageMarshaller () [inline, virtual]`

6.68.3 Member Function Documentation

6.68.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.68.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.68.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller::marshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2180).

6.68.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2180).

6.68.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2181).

6.68.3.6 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions

- IOException* if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2182).

```
6.68.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller::tightUnm
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2182).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQStreamMessageMarshaller.h`

6.69 activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMes

Class Reference

Marshaling code for Open Wire Format for `ActiveMQStreamMessageMarshaller` (p. 451).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQStreamMessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller`:

Public Member Functions

- **ActiveMQStreamMessageMarshaller** ()
- virtual **~ActiveMQStreamMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.69.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 451).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.69.2 Constructor & Destructor Documentation

6.69.2.1 `activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller::ActiveMQStreamMessageMarshaller () [inline]`

6.69.2.2 `virtual activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller::~ActiveMQStreamMessageMarshaller () [inline, virtual]`

6.69.3 Member Function Documentation

6.69.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.69.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.69.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller::marshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2167).

6.69.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2168).

6.69.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2169).

6.69.3.6 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions

- IOException* if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2169).

```
6.69.3.7 virtual void ac-
      tivemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller::tightUnm
      ( OpenWireFormat * wireFormat, commands::DataStructure
      * dataStructure, decaf::io::DataInputStream * dataIn,
      utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2170).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQStreamMessageMarshaller.h`

6.70 activemq::commands::ActiveMQTempDestination Class Reference

```
#include <src/main/activemq/commands/ActiveMQTempDestination.h>
```

Inheritance diagram for `activemq::commands::ActiveMQTempDestination`:

Public Member Functions

- **ActiveMQTempDestination** ()
- **ActiveMQTempDestination** (const std::string &name)
- virtual ~**ActiveMQTempDestination** ()
- virtual unsigned char **getDataStructureType** () const
- virtual **ActiveMQTempDestination** * **cloneDataStructure** () const

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)

Copy the contents of the passed object into this objects members, overwriting any existing data.

- virtual std::string **toString** () const

*Returns a string containing the information for this **DataStructure** (p. 1372) such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** *value) const

*Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent.*

- virtual void **close** () throw (cms::CMSException)

Closes down this Destination resulting in a call to dispose of the TempDestination resource at the Broker.

- void **setConnection** (core::ActiveMQConnection *connection)

Sets the Parent Connection that is notified when this destination is destroyed.

Static Public Attributes

- static const unsigned char **ID _ACTIVEMQTEMPDESTINATION** = 0

Protected Attributes

- core::ActiveMQConnection * **connection**

Connection that we call back on close to allow this resource to be cleaned up correctly at this end and at the Broker End.

6.70.1 Constructor & Destructor Documentation

- 6.70.1.1** `activemq::commands::ActiveMQTempDestination::ActiveMQTempDestination ()`
- 6.70.1.2** `activemq::commands::ActiveMQTempDestination::ActiveMQTempDestination (const std::string & name)`
- 6.70.1.3** `virtual
activemq::commands::ActiveMQTempDestination::~~ActiveMQTempDestination () [virtual]`

6.70.2 Member Function Documentation

- 6.70.2.1** `virtual ActiveMQTempDestination* activemq::commands::ActiveMQTempDestination::cloneDataStructure () const [inline, virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

- 6.70.2.2** `virtual void activemq::commands::ActiveMQTempDestination::close () throw (cms::CMSException) [virtual]`

Closes down this Destination resulting in a call to dispose of the TempDestination resource at the Broker.

This should only be called when the user is certain that they are finished with this destination. The TempDestination is not closed automatically on shutdown. throws **cms::CMSException** (p. 960)

Implements **cms::Closeable** (p. 952).

- 6.70.2.3** `virtual void activemq::commands::ActiveMQTempDestination::copyDataStructure (const DataStructure * src) [inline, virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

References `activemq::commands::ActiveMQDestination::copyDataStructure()`.

- 6.70.2.4** `virtual bool activemq::commands::ActiveMQTempDestination::equals (const DataStructure * value) const [inline, virtual]`

Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

References `activemq::commands::ActiveMQDestination::equals()`.

6.70.2.5 `virtual unsigned char activemq::commands::ActiveMQTempDestination::getDataStructureType () const [virtual]`

6.70.2.6 `void activemq::commands::ActiveMQTempDestination::setConnection (core::ActiveMQConnection * connection) [inline]`

Sets the Parent Connection that is notified when this destination is destroyed.

Parameters

connection - The parent connection.

6.70.2.7 `virtual std::string activemq::commands::ActiveMQTempDestination::toString () const [virtual]`

Returns a string containing the information for this **DataStructure** (p.1372) such as its type and value of its elements.

Returns

formatted string useful for debugging.

6.70.3 Field Documentation

6.70.3.1 `core::ActiveMQConnection* activemq::commands::ActiveMQTempDestination::connection [protected]`

Connection that we call back on close to allow this resource to be cleaned up correctly at this end and at the Broker End.

6.70.3.2 `const unsigned char activemq::commands::ActiveMQTempDestination::ID _ - ACTIVEMQTEMPDESTINATION = 0 [static]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ActiveMQTempDestination.h`

6.71

activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller

Class Reference

6.71 ~~activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller~~ ⁴⁶¹

Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 459).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTempDestinationMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller:

Public Member Functions

- **ActiveMQTempDestinationMarshaller** ()
- virtual **~ActiveMQTempDestinationMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.71.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 459).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.71.2 Constructor & Destructor Documentation

6.71.2.1 `activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller::ActiveMQTempDestinationMarshaller()` [inline]

6.71.2.2 `virtual activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller::~ActiveMQTempDestinationMarshaller()` [inline, virtual]

6.71.3 Member Function Documentation

6.71.3.1 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller::marshal(const OpenWireFormat * wireFormat, const commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut)` throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller` (p. 251).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller` (p. 483), and `activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller` (p. 507).

6.71.3.2 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller::unmarshal(const OpenWireFormat * wireFormat, const commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn)` throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

6.71

activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller

Class Reference

463

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller** (p. 252).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller** (p. 483), and **activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller** (p. 507).

```
6.71.3.3  virtual int ac-
          tivemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller::tightMa
          ( OpenWireFormat * wireFormat, commands::DataStructure *
            dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException
          ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller** (p. 252).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller** (p. 483), and **activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller** (p. 507).

```
6.71.3.4  virtual void ac-
          tivemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller::tightMa
          ( OpenWireFormat * wireFormat, commands::DataStructure
            * dataStructure, decaf::io::DataOutputStream * dataOut,
            utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller` (p. 253).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller` (p. 484), and `activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller` (p. 508).

```
6.71.3.5 virtual void ac-
      tivemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller::tightUn
      ( OpenWireFormat * wireFormat, commands::DataStructure
      * dataStructure, decaf::io::DataInputStream * dataIn,
      utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller` (p. 253).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller` (p. 484), and `activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller` (p. 508).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTempDestinationMarshaller.h`

6.72 activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestin Class Reference

Marshaling code for Open Wire Format for `ActiveMQTempDestinationMarshaller` (p. 462).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTempDestinationMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller`:

Public Member Functions

- `ActiveMQTempDestinationMarshaller ()`

6.72

activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller
Class Reference 465

-
- virtual `~ActiveMQTempDestinationMarshaller()`
 - virtual void **tightUnmarshal** (`OpenWireFormat *wireFormat`, `commands::DataStructure *dataStructure`, `decaf::io::DataInputStream *dataIn`, `utils::BooleanStream *bs`) throw (`decaf::io::IOException`)
Un-marshal an object instance from the data input stream.
 - virtual int **tightMarshal1** (`OpenWireFormat *wireFormat`, `commands::DataStructure *dataStructure`, `utils::BooleanStream *bs`) throw (`decaf::io::IOException`)
Write the booleans that this object uses to a BooleanStream.
 - virtual void **tightMarshal2** (`OpenWireFormat *wireFormat`, `commands::DataStructure *dataStructure`, `decaf::io::DataOutputStream *dataOut`, `utils::BooleanStream *bs`) throw (`decaf::io::IOException`)
Write a object instance to data output stream.
 - virtual void **looseUnmarshal** (`OpenWireFormat *wireFormat`, `commands::DataStructure *dataStructure`, `decaf::io::DataInputStream *dataIn`) throw (`decaf::io::IOException`)
Un-marshal an object instance from the data input stream.
 - virtual void **looseMarshal** (`OpenWireFormat *wireFormat`, `commands::DataStructure *dataStructure`, `decaf::io::DataOutputStream *dataOut`) throw (`decaf::io::IOException`)
Write a object instance to data output stream.

6.72.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 462).
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.72.2 Constructor & Destructor Documentation

- 6.72.2.1 **activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller::ActiveMQTempDestinationMarshaller()** [inline]
- 6.72.2.2 **virtual activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller::~ActiveMQTempDestinationMarshaller()** [inline, virtual]

6.72.3 Member Function Documentation

- 6.72.3.1 **virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller::looseMarshal1** (`OpenWireFormat * wireFormat`, `commands::DataStructure * dataStructure`, `decaf::io::DataOutputStream * dataOut`) throw (`decaf::io::IOException`) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller` (p. 255).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller` (p. 487), and `activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller` (p. 511).

6.72.3.2 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller` (p. 256).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller` (p. 487), and `activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller` (p. 511).

6.72.3.3 `virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled

6.72

activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller

Class Reference

467

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller` (p. 256).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller` (p. 487), and `activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller` (p. 511).

6.72.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller::tightMa`
(`OpenWireFormat * wireFormat`, `commands::DataStructure`
* `dataStructure`, `decaf::io::DataOutputStream * dataOut`,
`utils::BooleanStream * bs`) throw (`decaf::io::IOException`) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller` (p. 257).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller` (p. 488), and `activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller` (p. 512).

6.72.3.5 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller::tightUn`
(`OpenWireFormat * wireFormat`, `commands::DataStructure`
* `dataStructure`, `decaf::io::DataInputStream * dataIn`,
`utils::BooleanStream * bs`) throw (`decaf::io::IOException`) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller** (p. 257).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller** (p. 488), and **activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller** (p. 512).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTempDestinationMarshaller.h`

6.73 activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 466).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTempDestinationMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller**:

Public Member Functions

- **ActiveMQTempDestinationMarshaller** ()
- virtual **~ActiveMQTempDestinationMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

6.73

activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller
Class Reference **469**

- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.73.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 466).
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.73.2 Constructor & Destructor Documentation

6.73.2.1 **activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller::ActiveMQTempDestinationMarshaller** () [inline]

6.73.2.2 **virtual**
activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller::~ActiveMQTempDestinationMarshaller () [inline, virtual]

6.73.3 Member Function Documentation

6.73.3.1 **virtual void** **activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller::looseMarshal** (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller** (p. 259).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller** (p. 491), and **activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller** (p. 515).

```

6.73.3.2 virtual void ac-
    tivemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller::looseUn-
    ( OpenWireFormat * wireFormat, commands::DataStructure *
      dataStructure, decaf::io::DataInputStream * dataIn ) throw (
      decaf::io::IOException ) [virtual]

```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller` (p. 259).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller` (p. 491), and `activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller` (p. 515).

```

6.73.3.3 virtual int ac-
    tivemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller::tightMa-
    ( OpenWireFormat * wireFormat, commands::DataStructure *
      dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException
    ) [virtual]

```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller` (p. 260).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller` (p. 491), and `activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller` (p. 515).

6.73

activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller
Class Reference 471

```
6.73.3.4 virtual void ac-
        tivemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller::tightMa
        ( OpenWireFormat * wireFormat, commands::DataStructure
        * dataStructure, decaf::io::DataOutputStream * dataOut,
        utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller** (p. 260).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller** (p. 492), and **activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller** (p. 516).

```
6.73.3.5 virtual void ac-
        tivemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller::tightUnm
        ( OpenWireFormat * wireFormat, commands::DataStructure
        * dataStructure, decaf::io::DataInputStream * dataIn,
        utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller** (p. 261).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller** (p. 492), and **activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller** (p. 516).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**ActiveMQTempDestinationMarshaller.h**

6.74 activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 470).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTempDestinationMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller`:

Public Member Functions

- **ActiveMQTempDestinationMarshaller** ()
- virtual **~ActiveMQTempDestinationMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

6.74.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 470).

NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module

6.74.2 Constructor & Destructor Documentation

6.74.2.1 `activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller::ActiveMQTempDestinationMarshaller()` [inline]

6.74.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller::~ActiveMQTempDestinationMarshaller()` [inline, virtual]

6.74.3 Member Function Documentation

6.74.3.1 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller::marshal(const OpenWireFormat * wireFormat, const commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut)` throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller` (p. 263).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller` (p. 495), and `activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller` (p. 519).

6.74.3.2 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller::unmarshal(const OpenWireFormat * wireFormat, const commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn)` throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller` (p. 263).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller` (p. 495), and `activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller` (p. 519).

```
6.74.3.3  virtual int ac-
          tivemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller::tightMa
          ( OpenWireFormat * wireFormat, commands::DataStructure *
            dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException
          ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller` (p. 264).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller` (p. 495), and `activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller` (p. 519).

```
6.74.3.4  virtual void ac-
          tivemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller::tightMa
          ( OpenWireFormat * wireFormat, commands::DataStructure
            * dataStructure, decaf::io::DataOutputStream * dataOut,
            utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

6.75

activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller

Class Reference

475

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller** (p. 264).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller** (p. 496), and **activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller** (p. 520).

6.74.3.5 virtual void ac-

```
timemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller::tightUn  
( OpenWireFormat * wireFormat, commands::DataStructure  
* dataStructure, decaf::io::DataInputStream * dataIn,  
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller** (p. 265).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller** (p. 496), and **activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller** (p. 520).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**ActiveMQTempDestinationMarshaller.h**

6.75 activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestin

Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 473).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTempDestinationMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller**:

Public Member Functions

- **ActiveMQTempDestinationMarshaller** ()

- virtual ~**ActiveMQTempDestinationMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.75.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 473).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.75.2 Constructor & Destructor Documentation

- 6.75.2.1 **activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller::ActiveMQTempDestinationMarshaller** (**OpenWireFormat** *wireFormat) [**inline**]
- 6.75.2.2 **virtual** **activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller::~ActiveMQTempDestinationMarshaller** () [**inline**, **virtual**]

6.75.3 Member Function Documentation

- 6.75.3.1 **virtual void** **activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller::looseMarshal** (**OpenWireFormat** * wireFormat, **commands::DataStructure** * dataStructure, **decaf::io::DataOutputStream** * dataOut) throw (**decaf::io::IOException**) [**virtual**]

Write a object instance to data output stream.

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller** (p. 267).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller** (p. 499), and **activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller** (p. 523).

6.75.3.2 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller::looseUn-
(OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn) throw (
decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller** (p. 267).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller** (p. 499), and **activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller** (p. 523).

6.75.3.3 virtual int ac-
tivemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller::tightMa-
(OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException
) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller` (p. 268).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller` (p. 499), and `activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller` (p. 523).

```
6.75.3.4 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller::tightMa
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller` (p. 268).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller` (p. 500), and `activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller` (p. 524).

```
6.75.3.5 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller::tightUn
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller** (p. 269).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller** (p. 500), and **activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller** (p. 524).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTempDestinationMarshaller.h`

6.76 activemq::commands::ActiveMQTempQueue Class Reference

```
#include <src/main/activemq/commands/ActiveMQTempQueue.h>
```

Inheritance diagram for `activemq::commands::ActiveMQTempQueue`:

Public Member Functions

- **ActiveMQTempQueue** ()
- **ActiveMQTempQueue** (const std::string &name)
- virtual ~**ActiveMQTempQueue** ()
- virtual unsigned char **getDataStructureType** () const
- virtual **ActiveMQTempQueue** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
Converts the Destination Name into a String.
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent.*
- virtual const **cms::Destination** * **getCMSDestination** () const
- virtual **cms::Destination::DestinationType** **getDestinationType** () const
Retrieve the Destination Type for this Destination.

- virtual **cms::Destination * clone** () const
Creates a new instance of this destination type that is a copy of this one, and returns it.
- virtual void **copy** (const **cms::Destination** &source)
Copies the contents of the given Destination object to this one.
- virtual const **cms::CMSProperties** & **getCMSProperties** () const
Retrieve any properties that might be part of the destination that was specified.
- virtual std::string **getQueueName** () const throw (cms::CMSException)
Gets the name of this queue.
- virtual void **destroy** () throw (cms::CMSException)
Destroy's the Temp Destination at the Broker.

Static Public Attributes

- static const unsigned char **ID_ACTIVEMQTEMPQUEUE** = 102

6.76.1 Constructor & Destructor Documentation

- 6.76.1.1 **activemq::commands::ActiveMQTempQueue::ActiveMQTempQueue** ()
- 6.76.1.2 **activemq::commands::ActiveMQTempQueue::ActiveMQTempQueue** (const std::string & *name*)
- 6.76.1.3 **virtual**
activemq::commands::ActiveMQTempQueue::~~ActiveMQTempQueue () [virtual]

6.76.2 Member Function Documentation

- 6.76.2.1 **virtual cms::Destination*** **activemq::commands::ActiveMQTempQueue::clone** ()
const [inline, virtual]

Creates a new instance of this destination type that is a copy of this one, and returns it.

Returns

cloned copy of this object

Implements **cms::Destination** (p.1389).

- 6.76.2.2 **virtual ActiveMQTempQueue*** **activemq::commands::ActiveMQTempQueue::cloneDataStructure** () **const** [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

6.76.2.3 `virtual void activemq::commands::ActiveMQTempQueue::copy (const cms::Destination & source) [inline, virtual]`

Copies the contents of the given Destination object to this one.

Parameters

source The source Destination object.

Implements `cms::Destination` (p. 1389).

6.76.2.4 `virtual void activemq::commands::ActiveMQTempQueue::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

6.76.2.5 `virtual void activemq::commands::ActiveMQTempQueue::destroy () throw (cms::CMSException) [virtual]`

Destroy's the Temp Destination at the Broker.

Exceptions

CMSException

Implements `cms::TemporaryQueue` (p. 2972).

6.76.2.6 `virtual bool activemq::commands::ActiveMQTempQueue::equals (const DataStructure * value) const [virtual]`

Compares the `DataStructure` (p. 1372) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

6.76.2.7 `virtual const cms::Destination* activemq::commands::ActiveMQTempQueue::getCMSDestination () const`
`[inline, virtual]`

Returns

the **cms::Destination** (p.1387) interface pointer that the objects that derive from this class implement.

6.76.2.8 `virtual const cms::CMSProperties& activemq::commands::ActiveMQTempQueue::getCMSProperties () const`
`[inline, virtual]`

Retrieve any properties that might be part of the destination that was specified.

This is a deviation from the JMS spec but necessary due to C++ restrictions.

Returns

const reference to a properties object.

Implements **cms::Destination** (p.1389).

6.76.2.9 `virtual unsigned char activemq::commands::ActiveMQTempQueue::getDataStructureType () const`
`[virtual]`

6.76.2.10 `virtual cms::Destination::DestinationType activemq::commands::ActiveMQTempQueue::getDestinationType () const`
`[inline, virtual]`

Retrieve the Destination Type for this Destination.

Returns

The Destination Type

Implements **cms::Destination** (p.1389).

6.76.2.11 `virtual std::string activemq::commands::ActiveMQTempQueue::getQueueName () const throw (cms::CMSException)`
`[inline, virtual]`

Gets the name of this queue.

Returns

The queue name.

Implements **cms::TemporaryQueue** (p.2972).

6.76.2.12 `virtual std::string activemq::commands::ActiveMQTempQueue::toString
() const [virtual]`

Converts the Destination Name into a String.

Returns

string name

6.76.3 Field Documentation

6.76.3.1 `const unsigned char activemq::commands::ActiveMQTempQueue::ID_-
ACTIVEMQTEMPQUEUE = 102 [static]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ActiveMQTempQueue.h`

6.77 `activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller` Class Reference

Marshaling code for Open Wire Format for `ActiveMQTempQueueMarshaller` (p. 481).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTempQueueMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller`:

Public Member Functions

- `ActiveMQTempQueueMarshaller ()`
- `virtual ~ActiveMQTempQueueMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaler.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.77.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 481).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.77.2 Constructor & Destructor Documentation

- 6.77.2.1** **activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller::ActiveMQTempQueueMarshaller** () [inline]

- 6.77.2.2** **virtual**
activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller::~~ActiveMQTempQueueMarshaller () [inline, virtual]

6.77.3 Member Function Documentation

- 6.77.3.1** **virtual commands::DataStructure*** **activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller::createObject** () **const** [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1331).

- 6.77.3.2** **virtual unsigned char** **activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller::getDataStructureType** () **const** [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.77.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller` (p. 460).

6.77.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller` (p. 460).

6.77.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller` (p. 461).

```
6.77.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller::tightMarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller` (p. 461).

```
6.77.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller::tightUnmarsh
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller** (p. 462).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTempQueueMarshaller.h

6.78 activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 485).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTempQueueMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller**:

Public Member Functions

- **ActiveMQTempQueueMarshaller** ()
- virtual **~ActiveMQTempQueueMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

- virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.78.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 485).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.78.2 Constructor & Destructor Documentation

6.78.2.1 `activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller::ActiveMQTempQueueMarshaller () [inline]`

6.78.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller::~~ActiveMQTempQueueMarshaller () [inline, virtual]`

6.78.3 Member Function Documentation

6.78.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.78.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.78.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller` (p. 463).

6.78.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller` (p. 464).

6.78.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller` (p. 464).

```
6.78.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller::tightMarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller` (p. 465).

```
6.78.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller::tightUnmarsh
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException*** if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller** (p. 465).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTempQueueMarshaller.h

6.79 activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 489).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTempQueueMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller**:

Public Member Functions

- **ActiveMQTempQueueMarshaller** ()
- virtual **~ActiveMQTempQueueMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)

Un-marshall an object instance from the data input stream.

- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)`

Write a object instance to data output stream.

6.79.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 489).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.79.2 Constructor & Destructor Documentation

6.79.2.1 `activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller::ActiveMQTempQueueMarshaller () [inline]`

6.79.2.2 `virtual activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller::~~ActiveMQTempQueueMarshaller () [inline, virtual]`

6.79.3 Member Function Documentation

6.79.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1331).

6.79.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1336).

6.79.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller` (p. 467).

6.79.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller` (p. 468).

6.79.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller` (p. 468).

```
6.79.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller::tightMarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller` (p. 469).

```
6.79.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller::tightUnmarsh
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException*** if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller** (p. 469).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**ActiveMQTempQueueMarshaller.h**

6.80 activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 493).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTempQueueMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller**:

Public Member Functions

- **ActiveMQTempQueueMarshaller** ()
- virtual **~ActiveMQTempQueueMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)

Un-marshall an object instance from the data input stream.

- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)`

Write a object instance to data output stream.

6.80.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 493).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.80.2 Constructor & Destructor Documentation

6.80.2.1 `activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller::ActiveMQTempQueueMarshaller () [inline]`

6.80.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller::~~ActiveMQTempQueueMarshaller () [inline, virtual]`

6.80.3 Member Function Documentation

6.80.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1331).

6.80.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1336).

6.80.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller` (p. 471).

6.80.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller` (p. 471).

6.80.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller` (p. 472).

```
6.80.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller::tightMarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller` (p. 472).

```
6.80.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller::tightUnmarsh
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException*** if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller** (p. 473).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTempQueueMarshaller.h

6.81 activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 497).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTempQueueMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller**:

Public Member Functions

- **ActiveMQTempQueueMarshaller** ()
- virtual **~ActiveMQTempQueueMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshall an object instance from the data input stream.

- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)`

Write a object instance to data output stream.

6.81.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 497).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.81.2 Constructor & Destructor Documentation

6.81.2.1 `activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller::ActiveMQTempQueueMarshaller () [inline]`

6.81.2.2 `virtual activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller::~~ActiveMQTempQueueMarshaller () [inline, virtual]`

6.81.3 Member Function Documentation

6.81.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1331).

6.81.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1336).

6.81.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller` (p. 474).

6.81.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller` (p. 475).

6.81.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller` (p. 475).

```
6.81.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller::tightMarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller` (p. 476).

```
6.81.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller::tightUnmarsh
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller** (p. 476).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTempQueueMarshaller.h`

6.82 activemq::commands::ActiveMQTempTopic Class Reference

```
#include <src/main/activemq/commands/ActiveMQTempTopic.h>
```

Inheritance diagram for `activemq::commands::ActiveMQTempTopic`:

Public Member Functions

- **ActiveMQTempTopic** ()
- **ActiveMQTempTopic** (const std::string &name)
- virtual **~ActiveMQTempTopic** ()
- virtual unsigned char **getDataStructureType** () const
- virtual **ActiveMQTempTopic * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
Converts the Destination Name into a String.
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent.*
- virtual const **cms::Destination** * **getCMSDestination** () const
- virtual **cms::Destination::DestinationType** **getDestinationType** () const
Retrieve the Destination Type for this Destination.
- virtual **cms::Destination** * **clone** () const
Creates a new instance of this destination type that is a copy of this one, and returns it.
- virtual void **copy** (const **cms::Destination** &source)
Copies the contents of the given Destinastion object to this one.
- virtual const **cms::CMSProperties** & **getCMSProperties** () const

Retrieve any properties that might be part of the destination that was specified.

- virtual std::string **getTopicName** () const throw (cms::CMSEException)

Gets the name of this topic.

- virtual void **destroy** () throw (cms::CMSEException)

Destroy's the Temp Destination at the Broker.

Static Public Attributes

- static const unsigned char **ID_ACTIVEMQTEMPTOPIC** = 103

6.82.1 Constructor & Destructor Documentation

6.82.1.1 `activemq::commands::ActiveMQTempTopic::ActiveMQTempTopic ()`

6.82.1.2 `activemq::commands::ActiveMQTempTopic::ActiveMQTempTopic (const std::string & name)`

6.82.1.3 `virtual
activemq::commands::ActiveMQTempTopic::~~ActiveMQTempTopic ()
[virtual]`

6.82.2 Member Function Documentation

6.82.2.1 `virtual cms::Destination* ac-
tivismq::commands::ActiveMQTempTopic::clone ()
const [inline, virtual]`

Creates a new instance of this destination type that is a copy of this one, and returns it.

Returns

cloned copy of this object

Implements **cms::Destination** (p.1389).

6.82.2.2 `virtual ActiveMQTempTopic* ac-
tivismq::commands::ActiveMQTempTopic::cloneDataStructure () const
[virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

6.82.2.3 `virtual void activemq::commands::ActiveMQTempTopic::copy (const cms::Destination & source) [inline, virtual]`

Copies the contents of the given Destination object to this one.

Parameters

source The source Destination object.

Implements `cms::Destination` (p.1389).

6.82.2.4 `virtual void activemq::commands::ActiveMQTempTopic::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

6.82.2.5 `virtual void activemq::commands::ActiveMQTempTopic::destroy () throw (cms::CMSException) [virtual]`

Destroy's the Temp Destination at the Broker.

Exceptions

CMSException

Implements `cms::TemporaryTopic` (p.2973).

6.82.2.6 `virtual bool activemq::commands::ActiveMQTempTopic::equals (const DataStructure * value) const [virtual]`

Compares the `DataStructure` (p.1372) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

6.82.2.7 `virtual const cms::Destination* activemq::commands::ActiveMQTempTopic::getCMSDestination () const [inline, virtual]`

Returns

the `cms::Destination` (p.1387) interface pointer that the objects that derive from this class implement.

6.82.2.8 `virtual const cms::CMSProperties& activemq::commands::ActiveMQTempTopic::getCMSProperties () const`
[inline, virtual]

Retrieve any properties that might be part of the destination that was specified.

This is a deviation from the JMS spec but necessary due to C++ restrictions.

Returns

const reference to a properties object.

Implements **cms::Destination** (p.1389).

6.82.2.9 `virtual unsigned char activemq::commands::ActiveMQTempTopic::getDataStructureType () const`
[virtual]

6.82.2.10 `virtual cms::Destination::DestinationType activemq::commands::ActiveMQTempTopic::getDestinationType () const`
[inline, virtual]

Retrieve the Destination Type for this Destination.

Returns

The Destination Type

Implements **cms::Destination** (p.1389).

6.82.2.11 `virtual std::string activemq::commands::ActiveMQTempTopic::getTopicName () const throw (cms::CMSException)`
[inline, virtual]

Gets the name of this topic.

Returns

The topic name.

Implements **cms::TemporaryTopic** (p.2974).

6.82.2.12 `virtual std::string activemq::commands::ActiveMQTempTopic::toString () const`
[virtual]

Converts the Destination Name into a String.

Returns

string name

6.82.3 Field Documentation

- 6.82.3.1 `const unsigned char activemq::commands::ActiveMQTempTopic::ID _ -
 ACTIVEMQTEMPTOPIC = 103 [static]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ActiveMQTempTopic.h`

6.83 activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopic Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 505).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTempTopicMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller`:

Public Member Functions

- **ActiveMQTempTopicMarshaller** ()
- virtual **~ActiveMQTempTopicMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **com-
 mands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn,
utils::BooleanStream *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **com-
 mands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (de-
 caf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **com-
 mands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut,
utils::BooleanStream *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **com-
 mands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.

- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)`

Write a object instance to data output stream.

6.83.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 505). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.83.2 Constructor & Destructor Documentation

6.83.2.1 `activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller::ActiveMQTempTopicMarshaller () [inline]`

6.83.2.2 `virtual activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller::~~ActiveMQTempTopicMarshaller () [inline, virtual]`

6.83.3 Member Function Documentation

6.83.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1331).

6.83.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1336).

6.83.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller` (p. 460).

6.83.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller` (p. 460).

6.83.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller` (p. 461).

```
6.83.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller` (p. 461).

```
6.83.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller::tightUnmarsh
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller** (p. 462).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTempTopicMarshaller.h`

6.84 activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 509).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTempTopicMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller**:

Public Member Functions

- **ActiveMQTempTopicMarshaller** ()
- virtual **~ActiveMQTempTopicMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshall an object instance from the data input stream.

- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)`

Write a object instance to data output stream.

6.84.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 509). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.84.2 Constructor & Destructor Documentation

6.84.2.1 `activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller::ActiveMQTempTopicMarshaller () [inline]`

6.84.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller::~~ActiveMQTempTopicMarshaller () [inline, virtual]`

6.84.3 Member Function Documentation

6.84.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.84.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.84.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller` (p. 463).

6.84.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller` (p. 464).

6.84.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller` (p. 464).

```
6.84.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller` (p. 465).

```
6.84.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller::tightUnmarsh
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException*** if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller** (p. 465).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**ActiveMQTempTopicMarshaller.h**

6.85 activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 513).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTempTopicMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller**:

Public Member Functions

- **ActiveMQTempTopicMarshaller** ()
- virtual **~ActiveMQTempTopicMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshall an object instance from the data input stream.

- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)`

Write a object instance to data output stream.

6.85.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 513). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.85.2 Constructor & Destructor Documentation

6.85.2.1 `activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller::ActiveMQTempTopicMarshaller () [inline]`

6.85.2.2 `virtual activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller::~~ActiveMQTempTopicMarshaller () [inline, virtual]`

6.85.3 Member Function Documentation

6.85.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.85.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.85.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller` (p. 467).

6.85.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller` (p. 468).

6.85.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller` (p. 468).

```
6.85.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller` (p. 469).

```
6.85.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller::tightUnmarsh
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller` (p. 469).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTempTopicMarshaller.h`

6.86 activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller Class Reference

Marshaling code for Open Wire Format for `ActiveMQTempTopicMarshaller` (p. 517).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTempTopicMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller`:

Public Member Functions

- `ActiveMQTempTopicMarshaller ()`
- `virtual ~ActiveMQTempTopicMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaller.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write a object instance to data output stream.
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)`

Un-marshall an object instance from the data input stream.

- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)`

Write a object instance to data output stream.

6.86.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 517). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.86.2 Constructor & Destructor Documentation

6.86.2.1 `activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller::ActiveMQTempTopicMarshaller () [inline]`

6.86.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller::~~ActiveMQTempTopicMarshaller () [inline, virtual]`

6.86.3 Member Function Documentation

6.86.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.86.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.86.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller` (p. 471).

6.86.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller` (p. 471).

6.86.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller` (p. 472).

```
6.86.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller` (p. 472).

```
6.86.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller::tightUnmarsh
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException*** if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller** (p. 473).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**ActiveMQTempTopicMarshaller.h**

6.87 activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 521).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTempTopicMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller**:

Public Member Functions

- **ActiveMQTempTopicMarshaller** ()
- virtual **~ActiveMQTempTopicMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)

Un-marshall an object instance from the data input stream.

- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)`

Write a object instance to data output stream.

6.87.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 521). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.87.2 Constructor & Destructor Documentation

6.87.2.1 `activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller::ActiveMQTempTopicMarshaller () [inline]`

6.87.2.2 `virtual activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller::~ActiveMQTempTopicMarshaller () [inline, virtual]`

6.87.3 Member Function Documentation

6.87.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.87.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.87.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller` (p. 474).

6.87.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller` (p. 475).

6.87.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller` (p. 475).

```
6.87.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller` (p. 476).

```
6.87.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller::tightUnmarsh
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller** (p. 476).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTempTopicMarshaller.h`

6.88 activemq::commands::ActiveMQTextMessage Class Reference

```
#include <src/main/activemq/commands/ActiveMQTextMessage.h>
```

Inheritance diagram for `activemq::commands::ActiveMQTextMessage`:

Public Member Functions

- **ActiveMQTextMessage** ()
- virtual **~ActiveMQTextMessage** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaller share.
- virtual **ActiveMQTextMessage * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1372) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent.*
- virtual void **clearBody** () throw (cms::CMSException)
Clears out the body of the message.
- virtual void **beforeMarshal** (wireformat::WireFormat *wireFormat) throw (decaf::io::IOException)
*Performs any cleanup or other tasks that must be done before the **Message** (p. 2018) is marshalled to its binary WireFormat version.*
- virtual unsigned int **getSize** () const
Returns the Size of this message in Bytes.

- virtual **cms::TextMessage * clone** () const
Clone this message exactly, returns a new instance that the caller is required to delete.
- virtual std::string **getText** () const throw (cms::CMSEException)
Gets the message character buffer.
- virtual void **setText** (const char *msg) throw (cms::MessageNotWriteableException, cms::CMSEException)
Sets the message contents, does not take ownership of the passed char, but copies it instead.*
- virtual void **setText** (const std::string &msg) throw (cms::MessageNotWriteableException, cms::CMSEException)
Sets the message contents.

Data Fields

- std::auto_ptr< std::string > **text**

Static Public Attributes

- static const unsigned char **ID _ACTIVEMQTEXTMESSAGE** = 28

6.88.1 Constructor & Destructor Documentation

- 6.88.1.1 **activemq::commands::ActiveMQTextMessage::ActiveMQTextMessage** (
)
- 6.88.1.2 **virtual**
activemq::commands::ActiveMQTextMessage::~~ActiveMQTextMessage (
) [virtual]

6.88.2 Member Function Documentation

- 6.88.2.1 **virtual void activemq::commands::ActiveMQTextMessage::beforeMarshal**
(**wireformat::WireFormat * wireFormat**) throw (**decaf::io::IOException**
) [virtual]

Performs any cleanup or other tasks that must be done before the **Message** (p. 2018) is marshalled to its binary WireFormat version.

Parameters

wireFormat the WireFormat instance that is marshalling this message.

Implements **activemq::wireformat::MarshalAware** (p. 1994).

6.88.2.2 `virtual void activemq::commands::ActiveMQTextMessage::clearBody ()
throw (cms::CMSException) [virtual]`

Clears out the body of the message.

This does not clear the headers or properties.

Reimplemented from `activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 331).

6.88.2.3 `virtual cms::TextMessage* ac-
tivemq::commands::ActiveMQTextMessage::clone ()
const [inline, virtual]`

Clone this message exactly, returns a new instance that the caller is required to delete.

Returns

new copy of this message

6.88.2.4 `virtual ActiveMQTextMessage* ac-
tivemq::commands::ActiveMQTextMessage::cloneDataStructure ()
const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Reimplemented from `activemq::commands::Message` (p. 2023).

6.88.2.5 `virtual void ac-
tivemq::commands::ActiveMQTextMessage::copyDataStructure (const
DataStructure * src) [virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Reimplemented from `activemq::commands::Message` (p. 2023).

6.88.2.6 `virtual bool activemq::commands::ActiveMQTextMessage::equals (const
DataStructure * value) const [virtual]`

Compares the `DataStructure` (p. 1372) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::ActiveMQMessageTemplate<cms::TextMessage>` (p. 331).

6.88.2.7 `virtual unsigned char activemq::commands::ActiveMQTextMessage::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

Returns

new `DataStructure` (p. 1372) type copy.

Reimplemented from `activemq::commands::Message` (p. 2025).

6.88.2.8 `virtual unsigned int activemq::commands::ActiveMQTextMessage::getSize () const [virtual]`

Returns the Size of this message in Bytes.

Returns

number of bytes this message equates to.

Reimplemented from `activemq::commands::Message` (p. 2028).

6.88.2.9 `virtual std::string activemq::commands::ActiveMQTextMessage::getText () const throw (cms::CMSException) [virtual]`

Gets the message character buffer.

Returns

The message character buffer.

Exceptions

CMSException - if an internal error occurs.

6.88.2.10 `virtual void activemq::commands::ActiveMQTextMessage::setText (const std::string & msg) throw (cms::MessageNotWriteableException, cms::CMSException) [virtual]`

Sets the message contents.

Parameters

msg The message buffer.

Exceptions

CMSEException - if an internal error occurs.

MessageNotWriteableException - if the message is in read-only mode..

6.88.2.11 virtual void activemq::commands::ActiveMQTextMessage::setText
(const char * *msg*) throw (cms::MessageNotWriteableException,
cms::CMSEException) [virtual]

Sets the message contents, does not take ownership of the passed char*, but copies it instead.

Parameters

msg The message buffer.

Exceptions

CMSEException - if an internal error occurs.

MessageNotWriteableException - if the message is in read-only mode..

6.88.2.12 virtual std::string activemq::commands::ActiveMQTextMessage::toString
() const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1372) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::Message** (p. 2033).

6.88.3 Field Documentation

6.88.3.1 const unsigned char activemq::commands::ActiveMQTextMessage::ID_ -
ACTIVEMQTEXTMESSAGE = 28 [static]

6.88.3.2 std::auto_ptr<std::string> ac-
tivemq::commands::ActiveMQTextMessage::text
[mutable]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**ActiveMQTextMessage.h**

6.89 activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 529).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTextMessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller`:

Public Member Functions

- **ActiveMQTextMessageMarshaller** ()
- virtual **~ActiveMQTextMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.89.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 529).
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module

6.89.2 Constructor & Destructor Documentation

6.89.2.1 `activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller::ActiveMQTextMessageMarshaller () [inline]`

6.89.2.2 `virtual activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller::~~ActiveMQTextMessageMarshaller () [inline, virtual]`

6.89.3 Member Function Documentation

6.89.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.89.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.89.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2176).

6.89.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2176).

6.89.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2177).

6.89.3.6 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2178).

```
6.89.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller::tightUnmars
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2178).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTextMessageMarshaller.h`

6.90 activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller Class Reference

Marshaling code for Open Wire Format for `ActiveMQTextMessageMarshaller` (p. 533).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTextMessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller`:

Public Member Functions

- **ActiveMQTextMessageMarshaller** ()
- virtual **~ActiveMQTextMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.90.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p.533).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.90.2 Constructor & Destructor Documentation

6.90.2.1 `activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller::ActiveMQTextMessageMarshaller () [inline]`

6.90.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller::~~ActiveMQTextMessageMarshaller () [inline, virtual]`

6.90.3 Member Function Documentation

6.90.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.90.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.90.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2184).

6.90.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2184).

6.90.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2185).

6.90.3.6 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2186).

```
6.90.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller::tightUnmars
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2186).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTextMessageMarshaller.h`

6.91 `activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller` Class Reference

Marshaling code for Open Wire Format for `ActiveMQTextMessageMarshaller` (p. 537).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTextMessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller`:

Public Member Functions

- **ActiveMQTextMessageMarshaller** ()
- virtual **~ActiveMQTextMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.91.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p.537).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.91.2 Constructor & Destructor Documentation

6.91.2.1 `activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller::ActiveMQTextMessageMarshaller()` [inline]

6.91.2.2 `virtual activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller::~~ActiveMQTextMessageMarshaller()` [inline, virtual]

6.91.3 Member Function Documentation

6.91.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller::createObjectFromWireFormat(const commands::DataStructure&) const` [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.91.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.91.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller::looseMarshal(const OpenWireFormat* wireFormat, commands::DataStructure* dataStructure, decaf::io::DataOutputStream* dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2172).

6.91.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2172).

6.91.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2173).

6.91.3.6 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions

- IOException* if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2174).

```
6.91.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller::tightUnmars
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2174).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTextMessageMarshaller.h`

6.92 activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller Class Reference

Marshaling code for Open Wire Format for `ActiveMQTextMessageMarshaller` (p. 541).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTextMessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller`:

Public Member Functions

- **ActiveMQTextMessageMarshaller** ()
- virtual **~ActiveMQTextMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.92.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p.541).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.92.2 Constructor & Destructor Documentation

6.92.2.1 `activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller::ActiveMQTextMessageMarshaller () [inline]`

6.92.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller::~~ActiveMQTextMessageMarshaller () [inline, virtual]`

6.92.3 Member Function Documentation

6.92.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.92.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.92.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2180).

6.92.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2180).

6.92.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2181).

6.92.3.6 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2182).

```
6.92.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller::tightUnmars
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2182).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTextMessageMarshaller.h`

6.93 activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller Class Reference

Marshaling code for Open Wire Format for `ActiveMQTextMessageMarshaller` (p. 545).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTextMessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller`:

Public Member Functions

- **ActiveMQTextMessageMarshaller** ()
- virtual **~ActiveMQTextMessageMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.93.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p.545).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.93.2 Constructor & Destructor Documentation

6.93.2.1 `activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller::ActiveMQTextMessageMarshaller () [inline]`

6.93.2.2 `virtual activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller::~~ActiveMQTextMessageMarshaller () [inline, virtual]`

6.93.3 Member Function Documentation

6.93.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.93.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.93.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2167).

6.93.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2168).

6.93.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2169).

6.93.3.6 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions

- IOException* if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2169).

```
6.93.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller::tightUnmars
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2170).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTextMessageMarshaller.h`

6.94 activemq::commands::ActiveMQTopic Class Reference

```
#include <src/main/activemq/commands/ActiveMQTopic.h>
```

Inheritance diagram for `activemq::commands::ActiveMQTopic`:

Public Member Functions

- **ActiveMQTopic** ()
- **ActiveMQTopic** (const std::string &name)
- virtual ~**ActiveMQTopic** ()
- virtual unsigned char **getDataStructureType** () const
- virtual **ActiveMQTopic** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1372) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent.*
- virtual const cms::Destination * **getCMSDestination** () const
- virtual cms::Destination::DestinationType **getDestinationType** () const
Retrieve the Destination Type for this Destination.
- virtual cms::Destination * **clone** () const
Creates a new instance of this destination type that is a copy of this one, and returns it.
- virtual void **copy** (const cms::Destination &source)
Copies the contents of the given Destination object to this one.
- virtual const cms::CMSProperties & **getCMSProperties** () const
Retrieve any properties that might be part of the destination that was specified.
- virtual std::string **getTopicName** () const throw (cms::CMSException)
Gets the name of this topic.

Static Public Attributes

- static const unsigned char **ID_ACTIVEMQTOPIC** = 101

6.94.1 Constructor & Destructor Documentation

6.94.1.1 `activemq::commands::ActiveMQTopic::ActiveMQTopic ()`

6.94.1.2 `activemq::commands::ActiveMQTopic::ActiveMQTopic (const std::string & name)`

6.94.1.3 `virtual activemq::commands::ActiveMQTopic::~~ActiveMQTopic ()`
[virtual]

6.94.2 Member Function Documentation

6.94.2.1 `virtual cms::Destination* activemq::commands::ActiveMQTopic::clone () const` [inline, virtual]

Creates a new instance of this destination type that is a copy of this one, and returns it.

Returns

cloned copy of this object

Implements **cms::Destination** (p.1389).

6.94.2.2 `virtual ActiveMQTopic* activemq::commands::ActiveMQTopic::cloneDataStructure () const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

6.94.2.3 `virtual void activemq::commands::ActiveMQTopic::copy (const cms::Destination & source)` [inline, virtual]

Copies the contents of the given Destination object to this one.

Parameters

source The source Destination object.

Implements **cms::Destination** (p.1389).

6.94.2.4 `virtual void activemq::commands::ActiveMQTopic::copyDataStructure (const DataStructure * src)` [virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

6.94.2.5 `virtual bool activemq::commands::ActiveMQTopic::equals (const DataStructure * value) const` [inline, virtual]

Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

References `activemq::commands::ActiveMQDestination::equals()`.

6.94.2.6 `virtual const cms::Destination* activemq::commands::ActiveMQTopic::getCMSDestination () const` [inline, virtual]

Returns

the **cms::Destination** (p. 1387) interface pointer that the objects that derive from this class implement.

6.94.2.7 `virtual const cms::CMSProperties& activemq::commands::ActiveMQTopic::getCMSProperties () const` [inline, virtual]

Retrieve any properties that might be part of the destination that was specified. This is a deviation from the JMS spec but necessary due to C++ restrictions.

Returns

const reference to a properties object.

Implements **cms::Destination** (p. 1389).

6.94.2.8 `virtual unsigned char activemq::commands::ActiveMQTopic::getDataStructureType () const` [virtual]

6.94.2.9 `virtual cms::Destination::DestinationType activemq::commands::ActiveMQTopic::getDestinationType () const` [inline, virtual]

Retrieve the Destination Type for this Destination.

Returns

The Destination Type

Implements **cms::Destination** (p. 1389).

6.94.2.10 `virtual std::string activemq::commands::ActiveMQTopic::getTopicName () const throw (cms::CMSException) [inline, virtual]`

Gets the name of this topic.

Returns

The topic name.

Implements **cms::Topic** (p. 3014).

6.94.2.11 `virtual std::string activemq::commands::ActiveMQTopic::toString () const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1372) such as its type and value of its elements.

Returns

formatted string useful for debugging.

6.94.3 Field Documentation

6.94.3.1 `const unsigned char activemq::commands::ActiveMQTopic::ID_ - ACTIVEMQTOPIC = 101 [static]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ActiveMQTopic.h`

6.95 activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 553).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTopicMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller`:

Public Member Functions

- **ActiveMQTopicMarshaller** ()
- virtual **~ActiveMQTopicMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.95.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 553). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.95.2 Constructor & Destructor Documentation

6.95.2.1 `activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller::ActiveMQTopicMarshaller () [inline]`

6.95.2.2 `virtual activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller::~~ActiveMQTopicMarshaller () [inline, virtual]`

6.95.3 Member Function Documentation

6.95.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1331).

6.95.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller::getDataStructureType() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1336).

6.95.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller** (p. 251).

6.95.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller` (p. 252).

```
6.95.3.5 virtual int ac-
      tivemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller::tightMarshal1
      ( OpenWireFormat * wireFormat, commands::DataStructure *
        dataStructure, utils::BooleanStream * bs ) throw ( decaf::io::IOException
      ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller` (p. 252).

```
6.95.3.6 virtual void ac-
      tivemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller::tightMarshal2
      ( OpenWireFormat * wireFormat, commands::DataStructure
        * dataStructure, decaf::io::DataOutputStream * dataOut,
        utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller` (p. 253).

```
6.95.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller** (p. 253).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTopicMarshaller.h

6.96 activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 557).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTopicMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller**:

Public Member Functions

- **ActiveMQTopicMarshaller** ()
- virtual **~ActiveMQTopicMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.96.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 557). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.96.2 Constructor & Destructor Documentation

6.96.2.1 `activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller::ActiveMQTopicMarshaller () [inline]`

6.96.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller::~~ActiveMQTopicMarshaller () [inline, virtual]`

6.96.3 Member Function Documentation

6.96.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.96.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller::getDataStructureType() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.96.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller` (p. 255).

6.96.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller` (p. 256).

6.96.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller` (p. 256).

6.96.3.6 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller` (p. 257).

6.96.3.7 `virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller` (p. 257).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTopicMarshaller.h`

6.97 activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller Class Reference

Marshaling code for Open Wire Format for `ActiveMQTopicMarshaller` (p. 561).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTopicMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller`:

Public Member Functions

- `ActiveMQTopicMarshaller ()`
- `virtual ~ActiveMQTopicMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaler.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshall an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.97.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 561). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.97.2 Constructor & Destructor Documentation

6.97.2.1 **activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller::ActiveMQTopicMarshaller** () [inline]

6.97.2.2 **virtual** **activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller::~~ActiveMQTopicMarshaller** () [inline, virtual]

6.97.3 Member Function Documentation

6.97.3.1 **virtual commands::DataStructure*** **activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller::createObject** () **const** [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1331).

6.97.3.2 **virtual unsigned char** **activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller::getDataStructureType** () **const** [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.97.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller` (p. 259).

6.97.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller` (p. 259).

6.97.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller` (p. 260).

```
6.97.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller` (p. 260).

```
6.97.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller** (p. 261).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**ActiveMQTopicMarshaller.h**

6.98 activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 565).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTopicMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller**:

Public Member Functions

- **ActiveMQTopicMarshaller** ()
- virtual **~ActiveMQTopicMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

- virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.98.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 565). NOTE! This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.98.2 Constructor & Destructor Documentation

6.98.2.1 `activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller::ActiveMQTopicMarshaller () [inline]`

6.98.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller::~~ActiveMQTopicMarshaller () [inline, virtual]`

6.98.3 Member Function Documentation

6.98.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.98.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.98.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller` (p. 263).

6.98.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller` (p. 263).

6.98.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller` (p. 264).

```
6.98.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller` (p. 264).

```
6.98.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException*** if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller** (p. 265).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**ActiveMQTopicMarshaller.h**

6.99 activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller Class Reference

Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 569).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTopicMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller**:

Public Member Functions

- **ActiveMQTopicMarshaller** ()
- virtual **~ActiveMQTopicMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)

Un-marshall an object instance from the data input stream.

- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)`

Write a object instance to data output stream.

6.99.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 569). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.99.2 Constructor & Destructor Documentation

6.99.2.1 `activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller::ActiveMQTopicMarshaller () [inline]`

6.99.2.2 `virtual activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller::~~ActiveMQTopicMarshaller () [inline, virtual]`

6.99.3 Member Function Documentation

6.99.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.99.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.99.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller` (p. 267).

6.99.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller` (p. 267).

6.99.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller` (p. 268).

```
6.99.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller` (p. 268).

```
6.99.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller** (p. 269).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTopicMarshaller.h`

6.100 activemq::core::ActiveMQTransactionContext Class Reference

Transaction Management class, hold messages that are to be redelivered upon a request to roll-back.

```
#include <src/main/activemq/core/ActiveMQTransactionContext.h>
```

Public Member Functions

- **ActiveMQTransactionContext** (**ActiveMQSession** *session, const **decaf::util::Properties** &properties)
Constructor.
- virtual **~ActiveMQTransactionContext** ()
- virtual void **addSynchronization** (const **Pointer**< **Synchronization** > &sync)
*Adds a **Synchronization** (p. 2945) to this Transaction.*
- virtual void **removeSynchronization** (const **Pointer**< **Synchronization** > &sync)
*Removes a **Synchronization** (p. 2945) to this Transaction.*
- virtual void **begin** () throw (exceptions::ActiveMQException)
Begins a new transaction if one is not currently in progress.
- virtual void **commit** () throw (exceptions::ActiveMQException)
Commit the current Transaction.
- virtual void **rollback** () throw (exceptions::ActiveMQException)
Rollback the current Transaction.
- virtual const **decaf::lang::Pointer**< **commands::TransactionId** > & **getTransactionId** () const
Get the Transaction Id object for the current Transaction, returns NULL if no transaction is running.
- virtual bool **isInTransaction** () const
Checks to see if there is currently a Transaction in progress returns false if not, true otherwise.
- virtual int **getMaximumRedeliveries** () const
- virtual long long **getRedeliveryDelay** () const

6.100.1 Detailed Description

Transaction Management class, hold messages that are to be redelivered upon a request to roll-back. The Transaction represents an always running transaction, when it is committed or rolled back it silently creates a new transaction for the next set of messages. The only way to permanently end this transaction is to delete it.

Configuration options

`transaction.maxRedeliveryCount` Max number of times a message can be re-delivered, if the session is rolled back more than this many time, the message is dropped.

`transaction.redeliveryDelay` Time in Milliseconds between message redelivery for rolled back transactions.

Since

2.0

6.100.2 Constructor & Destructor Documentation

6.100.2.1 `activemq::core::ActiveMQTransactionContext::ActiveMQTransactionContext (ActiveMQSession * session, const decaf::util::Properties & properties)`

Constructor.

Parameters

session The session that contains this transaction

properties Configuration parameters for this object

6.100.2.2 `virtual
activemq::core::ActiveMQTransactionContext::~~ActiveMQTransactionContext
() [virtual]`

6.100.3 Member Function Documentation

6.100.3.1 `virtual void ac-
tivemq::core::ActiveMQTransactionContext::addSynchronization (const
Pointer< Synchronization > & sync) [virtual]`

Adds a **Synchronization** (p. 2945) to this Transaction.

Parameters

sync - The **Synchronization** (p. 2945) instance to add.

6.100.3.2 `virtual void activemq::core::ActiveMQTransactionContext::begin ()
throw (exceptions::ActiveMQException) [virtual]`

Begins a new transaction if one is not currently in progress.

Exceptions*ActiveMQException*

6.100.3.3 virtual void activemq::core::ActiveMQTransactionContext::commit ()
throw (exceptions::ActiveMQException) [virtual]

Commit the current Transaction.

Exceptions*ActiveMQException*

6.100.3.4 virtual int activemq::core::ActiveMQTransactionContext::getMaximumRedeliveries () const [virtual]

Returns

The Maximum number of time the client will attempt to redeliver a message from a rolled back transaction before marking the message as not consumed by this client.

6.100.3.5 virtual long long activemq::core::ActiveMQTransactionContext::getRedeliveryDelay () const [virtual]

Returns

The time in Milliseconds that this client is configured to wait in between redelivery attempts for a Message in a rolled back transaction.

6.100.3.6 virtual const decaf::lang::Pointer<commands::TransactionId>& activemq::core::ActiveMQTransactionContext::getTransactionId () const [virtual]

Get the Transaction Id object for the current Transaction, returns NULL if no transaction is running.

Returns

TransactionInfo

Exceptions

InvalidStateException if a Transaction is not in progress.

6.100.3.7 virtual bool activemq::core::ActiveMQTransactionContext::isInTransaction () const [virtual]

Checks to see if there is currently a Transaction in progress returns false if not, true otherwise.

Returns

true if a transaction is in progress.

6.100.3.8 virtual void activemq::core::ActiveMQTransactionContext::removeSynchronization (const Pointer< Synchronization > & *sync*) [virtual]

Removes a **Synchronization** (p. 2945) to this Transaction.

Parameters

sync - The **Synchronization** (p. 2945) instance to add.

6.100.3.9 virtual void activemq::core::ActiveMQTransactionContext::rollback () throw (exceptions::ActiveMQException) [virtual]

Rollback the current Transaction.

Exceptions

ActiveMQException

The documentation for this class was generated from the following file:

- src/main/activemq/core/ActiveMQTransactionContext.h

6.101 decaf::lang::Appendable Class Reference

```
#include <src/main/decaf/lang/Appendable.h>
```

Inheritance diagram for decaf::lang::Appendable:

Public Member Functions

- virtual ~**Appendable** ()
- virtual **Appendable** & **append** (char value)=0 throw (decaf::lang::Exception)
*Appends the specified character to this **Appendable** (p. 576).*
- virtual **Appendable** & **append** (const **CharSequence** *csq)=0 throw (decaf::lang::Exception)
*Appends the specified character sequence to this **Appendable** (p. 576).*
- virtual **Appendable** & **append** (const **CharSequence** *csq, std::size_t start, std::size_t end)=0 throw (decaf::lang::Exception)
*Appends a subsequence of the specified character sequence to this **Appendable** (p. 576).*

6.101.1 Constructor & Destructor Documentation

6.101.1.1 virtual decaf::lang::Appendable::~~Appendable () [inline, virtual]

6.101.2 Member Function Documentation

6.101.2.1 virtual Appendable& decaf::lang::Appendable::append (char *value*)
throw (decaf::lang::Exception) [pure virtual]

Appends the specified character to this **Appendable** (p. 576).

Parameters

value - The character to append

Returns

a Reference to this **Appendable** (p. 576)

Exceptions

Exception (p. 1476) if an error occurs.

6.101.2.2 virtual Appendable& decaf::lang::Appendable::append (const
CharSequence * *csq*, std::size_t *start*, std::size_t *end*) throw (decaf::lang::Exception) [pure virtual]

Appends a subsequence of the specified character sequence to this **Appendable** (p. 576).

Parameters

csq - The character sequence from which a subsequence will be appended. If *csq* is NULL, then characters will be appended as if *csq* contained the string "null".

start - The index of the first character in the subsequence

end - The index of the character following the last character in the subsequence

Returns

a Reference to this **Appendable** (p. 576)

Exceptions

Exception (p. 1476) if an error occurs.

IndexOutOfBoundsException *start* is greater than *end*, or *end* is greater than *csq.length()*

6.101.2.3 virtual Appendable& decaf::lang::Appendable::append (const
CharSequence * *csq*) throw (decaf::lang::Exception) [pure virtual]

Appends the specified character sequence to this **Appendable** (p. 576).

Parameters

csq - The character sequence from which a subsequence will be appended. If *csq* is NULL, then characters will be appended as if *csq* contained the string "null".

Returns

- a Reference to this **Appendable** (p. 576)

Exceptions

- Exception* (p. 1476) if an error occurs.

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Appendable.h`

6.102 decaf::internal::AprPool Class Reference

Wraps an APR pool object so that classes in decaf can create a static member for use in static methods where apr function calls that need a pool are made.

```
#include <src/main/decaf/internal/AprPool.h>
```

Public Member Functions

- **AprPool** ()
- virtual **~AprPool** ()
- `apr_pool_t * getAprPool () const`
Gets the internal APR Pool.
- void **cleanup** ()
Clears data that was allocated by this pool.

Static Public Member Functions

- `static apr_pool_t * getGlobalPool ()`
Gets a pointer to the Global APR Pool used for the Application.

6.102.1 Detailed Description

Wraps an APR pool object so that classes in decaf can create a static member for use in static methods where apr function calls that need a pool are made.

6.102.2 Constructor & Destructor Documentation

6.102.2.1 decaf::internal::AprPool::AprPool ()

6.102.2.2 virtual decaf::internal::AprPool::~~AprPool () [virtual]

6.102.3 Member Function Documentation

6.102.3.1 void decaf::internal::AprPool::cleanup ()

Clears data that was allocated by this pool.

Users should call this after getting the data from the APR functions and copying it to someplace safe.

6.102.3.2 apr_pool_t* decaf::internal::AprPool::getAprPool () const

Gets the internal APR Pool.

Returns

the internal APR pool

6.102.3.3 static apr_pool_t* decaf::internal::AprPool::getGlobalPool ()
[static]

Gets a pointer to the Global APR Pool used for the Application.

Returns

pointer to the global APR Pool

The documentation for this class was generated from the following file:

- src/main/decaf/internal/**AprPool.h**

6.103 decaf::util::concurrent::atomic::AtomicBoolean Class Reference

A boolean value that may be updated atomically.

```
#include <src/main/decaf/util/concurrent/atomic/AtomicBoolean.h>
```

Public Member Functions

- **AtomicBoolean** ()
*Creates a new **AtomicBoolean** (p. 579) whose initial value is false.*
- **AtomicBoolean** (bool initialValue)
*Creates a new **AtomicBoolean** (p. 579) with the initial value.*

- virtual `~AtomicBoolean ()`
- bool `get () const`
*Gets the current value of this **AtomicBoolean** (p. 579).*
- void `set (bool newValue)`
Unconditionally sets to the given value.
- bool `compareAndSet (bool expect, bool update)`
Atomically sets the value to the given updated value if the current value == the expected value.
- bool `getAndSet (bool newValue)`
Atomically sets to the given value and returns the previous value.
- std::string `toString () const`
Returns the String representation of the current value.

6.103.1 Detailed Description

A boolean value that may be updated atomically. An **AtomicBoolean** (p. 579) is used in applications such as atomically updated flags, and cannot be used as a replacement for a Boolean.

6.103.2 Constructor & Destructor Documentation

6.103.2.1 `decaf::util::concurrent::atomic::AtomicBoolean::AtomicBoolean ()`

Creates a new **AtomicBoolean** (p. 579) whose initial value is false.

6.103.2.2 `decaf::util::concurrent::atomic::AtomicBoolean::AtomicBoolean (bool initialValue)`

Creates a new **AtomicBoolean** (p. 579) with the initial value.

Parameters

initialValue - The initial value of this boolean.

6.103.2.3 `virtual decaf::util::concurrent::atomic::AtomicBoolean::~~AtomicBoolean () [inline, virtual]`

6.103.3 Member Function Documentation

6.103.3.1 `bool decaf::util::concurrent::atomic::AtomicBoolean::compareAndSet (bool expect, bool update)`

Atomically sets the value to the given updated value if the current value == the expected value.

Parameters

expect - the expected value

update - the new value

Returns

true if successful. False return indicates that the actual value was not equal to the expected value.

6.103.3.2 `bool decaf::util::concurrent::atomic::AtomicBoolean::get () const` [inline]

Gets the current value of this **AtomicBoolean** (p.579).

Returns

the currently set value.

6.103.3.3 `bool decaf::util::concurrent::atomic::AtomicBoolean::getAndSet (bool new Value)`

Atomically sets to the given value and returns the previous value.

Parameters

new Value - the new value

Returns

the previous value

6.103.3.4 `void decaf::util::concurrent::atomic::AtomicBoolean::set (bool new Value)` [inline]

Unconditionally sets to the given value.

Parameters

new Value - the new value

6.103.3.5 `std::string decaf::util::concurrent::atomic::AtomicBoolean::toString () const`

Returns the String representation of the current value.

Returns

the String representation of the current value.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/atomic/AtomicBoolean.h`

6.104 decaf::util::concurrent::atomic::AtomicInteger Class Reference

An int value that may be updated atomically.

```
#include <src/main/decaf/util/concurrent/atomic/AtomicInteger.h>
```

Inheritance diagram for decaf::util::concurrent::atomic::AtomicInteger:

Public Member Functions

- **AtomicInteger** ()
*Create a new **AtomicInteger** (p. 582) with an initial value of 0.*
- **AtomicInteger** (int initialValue)
*Create a new **AtomicInteger** (p. 582) with the given initial value.*
- virtual **~AtomicInteger** ()
- int **get** () const
Gets the current value.
- void **set** (int newValue)
Sets to the given value.
- int **getAndSet** (int newValue)
Atomically sets to the given value and returns the old value.
- bool **compareAndSet** (int expect, int update)
Atomically sets the value to the given updated value if the current value == the expected value.
- int **getAndIncrement** ()
Atomically increments by one the current value.
- int **getAndDecrement** ()
Atomically decrements by one the current value.
- int **getAndAdd** (int delta)
Atomically adds the given value to the current value.
- int **incrementAndGet** ()
Atomically increments by one the current value.
- int **decrementAndGet** ()
Atomically decrements by one the current value.
- int **addAndGet** (int delta)
Atomically adds the given value to the current value.

- `std::string toString () const`
Returns the String representation of the current value.
- `int intValue () const`
Description copied from class: Number Returns the value of the specified number as an int.
- `long long longValue () const`
Description copied from class: Number Returns the value of the specified number as a long.
- `float floatValue () const`
Description copied from class: Number Returns the value of the specified number as a float.
- `double doubleValue () const`
Description copied from class: Number Returns the value of the specified number as a double.

6.104.1 Detailed Description

An int value that may be updated atomically. An **AtomicInteger** (p. 582) is used in applications such as atomically incremented counters, and cannot be used as a replacement for an Integer. However, this class does extend Number to allow uniform access by tools and utilities that deal with numerically-based classes.

6.104.2 Constructor & Destructor Documentation

6.104.2.1 decaf::util::concurrent::atomic::AtomicInteger::AtomicInteger ()

Create a new **AtomicInteger** (p. 582) with an initial value of 0.

6.104.2.2 decaf::util::concurrent::atomic::AtomicInteger::AtomicInteger (int *initialValue*)

Create a new **AtomicInteger** (p. 582) with the given initial value.

Parameters

initialValue - The initial value of this object.

6.104.2.3 virtual decaf::util::concurrent::atomic::AtomicInteger::~~AtomicInteger () [inline, virtual]

6.104.3 Member Function Documentation

6.104.3.1 int decaf::util::concurrent::atomic::AtomicInteger::addAndGet (int *delta*)

Atomically adds the given value to the current value.

Parameters

delta - the value to add.

Returns

the updated value.

6.104.3.2 `bool decaf::util::concurrent::atomic::AtomicInteger::compareAndSet (int expect, int update)`

Atomically sets the value to the given updated value if the current value == the expected value.

Parameters

expect - the expected value

update - the new value

Returns

true if successful. False return indicates that the actual value was not equal to the expected value.

6.104.3.3 `int decaf::util::concurrent::atomic::AtomicInteger::decrementAndGet ()`

Atomically decrements by one the current value.

Returns

the updated value.

Referenced by `decaf::lang::AtomicRefCounter::release()`.

6.104.3.4 `double decaf::util::concurrent::atomic::AtomicInteger::doubleValue () const [virtual]`

Description copied from class: `Number` Returns the value of the specified number as a double.

This may involve rounding.

Returns

the numeric value represented by this object after conversion to type double.

Implements `decaf::lang::Number` (p. 2283).

6.104.3.5 `float decaf::util::concurrent::atomic::AtomicInteger::floatValue () const [virtual]`

Description copied from class: `Number` Returns the value of the specified number as a float.

This may involve rounding.

Returns

the numeric value represented by this object after conversion to type float.

Implements **decaf::lang::Number** (p. 2283).

6.104.3.6 `int decaf::util::concurrent::atomic::AtomicInteger::get () const`
[inline]

Gets the current value.

Returns

the current value.

6.104.3.7 `int decaf::util::concurrent::atomic::AtomicInteger::getAndAdd (int
delta)`

Atomically adds the given value to the current value.

Parameters

delta - The value to add.

Returns

the previous value.

6.104.3.8 `int decaf::util::concurrent::atomic::AtomicInteger::getAndDecrement ()`

Atomically decrements by one the current value.

Returns

the previous value.

6.104.3.9 `int decaf::util::concurrent::atomic::AtomicInteger::getAndIncrement ()`

Atomically increments by one the current value.

Returns

the previous value.

6.104.3.10 `int decaf::util::concurrent::atomic::AtomicInteger::getAndSet (int
new Value)`

Atomically sets to the given value and returns the old value.

Parameters

new Value - the new value.

Returns

the previous value.

6.104.3.11 `int decaf::util::concurrent::atomic::AtomicInteger::incrementAndGet ()`

Atomically increments by one the current value.

Returns

the updated value.

Referenced by `decaf::lang::AtomicRefCounter::AtomicRefCounter()`.

6.104.3.12 `int decaf::util::concurrent::atomic::AtomicInteger::intValue () const`
[virtual]

Description copied from class: Number Returns the value of the specified number as an int.

This may involve rounding or truncation.

Returns

the numeric value represented by this object after conversion to type int.

Implements **decaf::lang::Number** (p. 2283).

6.104.3.13 `long long decaf::util::concurrent::atomic::AtomicInteger::longValue ()`
`const` [virtual]

Description copied from class: Number Returns the value of the specified number as a long.

This may involve rounding or truncation.

Returns

the numeric value represented by this object after conversion to type long long.

Implements **decaf::lang::Number** (p. 2283).

6.104.3.14 `void decaf::util::concurrent::atomic::AtomicInteger::set (int new Value`
`)` [inline]

Sets to the given value.

Parameters

new Value - the new value

6.104.3.15 std::string decaf::util::concurrent::atomic::AtomicInteger::toString () const

Returns the String representation of the current value.

Returns

the String representation of the current value.

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/atomic/**AtomicInteger.h**

6.105 decaf::lang::AtomicRefCounter Class Reference

```
#include <src/main/decaf/lang/Pointer.h>
```

Inherited by decaf::lang::Pointer< activemq::threads::TaskRunner >, decaf::lang::Pointer< ActiveMQDestination >, decaf::lang::Pointer< ActiveMQTransactionContext >, decaf::lang::Pointer< BackupTransportPool >, decaf::lang::Pointer< BooleanExpression >, decaf::lang::Pointer< BrokerError >, decaf::lang::Pointer< BrokerId >, decaf::lang::Pointer< CloseTransportsTask >, decaf::lang::Pointer< cms::Destination >, decaf::lang::Pointer< commands::BrokerInfo >, decaf::lang::Pointer< commands::ConnectionInfo >, decaf::lang::Pointer< commands::ConsumerId >, decaf::lang::Pointer< commands::ConsumerInfo >, decaf::lang::Pointer< commands::Message >, decaf::lang::Pointer< commands::MessageAck >, decaf::lang::Pointer< commands::ProducerInfo >, decaf::lang::Pointer< commands::SessionInfo >, decaf::lang::Pointer< commands::TransactionId >, decaf::lang::Pointer< commands::WireFormatInfo >, decaf::lang::Pointer< Comparator< E > >, decaf::lang::Pointer< CompositeTaskRunner >, decaf::lang::Pointer< ConnectionId >, decaf::lang::Pointer< ConnectionInfo >, decaf::lang::Pointer< ConsumerId >, decaf::lang::Pointer< ConsumerInfo >, decaf::lang::Pointer< core::ActiveMQAckHandler >, decaf::lang::Pointer< DataStructure >, decaf::lang::Pointer< decaf::lang::Runnable >, decaf::lang::Pointer< decaf::lang::Thread >, decaf::lang::Pointer< decaf::util::Properties >, decaf::lang::Pointer< Exception >, decaf::lang::Pointer< InactivityMonitorData >, decaf::lang::Pointer< LockHandle >, decaf::lang::Pointer< Message >, decaf::lang::Pointer< MessageAck >, decaf::lang::Pointer< MessageId >, decaf::lang::Pointer< ProducerId >, decaf::lang::Pointer< ProducerInfo >, decaf::lang::Pointer< Response >, decaf::lang::Pointer< ResponseBuilder >, decaf::lang::Pointer< SessionId >, decaf::lang::Pointer< SessionInfo >, decaf::lang::Pointer< Tracked >, decaf::lang::Pointer< TransactionId >, decaf::lang::Pointer< Transport >, decaf::lang::Pointer< transport::Transport >, decaf::lang::Pointer< TransportListener >, decaf::lang::Pointer< URI >, decaf::lang::Pointer< URIPool >, and decaf::lang::Pointer< wireformat::WireFormat >.

Public Member Functions

- AtomicRefCounter ()
- AtomicRefCounter (const AtomicRefCounter &other)

Protected Member Functions

- void **swap** (**AtomicRefCounter** &other)

Swaps this instance's reference counter with the one given, this allows for copy-and-swap semantics of this object.

- bool **release** ()

Removes a reference to the counter Atomically and returns if the counter has reached zero, once the counter hits zero, the internal counter is destroyed and this instance is now considered to be unreferenced.

6.105.1 Constructor & Destructor Documentation

6.105.1.1 `decaf::lang::AtomicRefCounter::AtomicRefCounter ()` [inline]

6.105.1.2 `decaf::lang::AtomicRefCounter::AtomicRefCounter (const AtomicRefCounter & other)` [inline]

References `decaf::util::concurrent::atomic::AtomicInteger::incrementAndGet()`.

6.105.2 Member Function Documentation

6.105.2.1 `bool decaf::lang::AtomicRefCounter::release ()` [inline, protected]

Removes a reference to the counter Atomically and returns if the counter has reached zero, once the counter hits zero, the internal counter is destroyed and this instance is now considered to be unreferenced.

Returns

true if the count is now zero.

Reimplemented in `decaf::lang::Pointer< commands::ConsumerId >` (p. 2349), `decaf::lang::Pointer< MessageAck >` (p. 2349), `decaf::lang::Pointer< BooleanExpression >` (p. 2349), `decaf::lang::Pointer< BrokerError >` (p. 2349), `decaf::lang::Pointer< Transport >` (p. 2349), `decaf::lang::Pointer< wireformat::WireFormat >` (p. 2349), `decaf::lang::Pointer< CloseTransportsTask >` (p. 2349), `decaf::lang::Pointer< CompositeTaskRunner >` (p. 2349), `decaf::lang::Pointer< commands::WireFormatInfo >` (p. 2349), `decaf::lang::Pointer< ActiveMQTransactionContext >` (p. 2349), `decaf::lang::Pointer< commands::ProducerInfo >` (p. 2349), `decaf::lang::Pointer< Comparator< E > >` (p. 2349), `decaf::lang::Pointer< BrokerId >` (p. 2349), `decaf::lang::Pointer< commands::SessionInfo >` (p. 2349), `decaf::lang::Pointer< Message >` (p. 2349), `decaf::lang::Pointer< URI >` (p. 2349), `decaf::lang::Pointer< DataStructure >` (p. 2349), `decaf::lang::Pointer< commands::ConnectionInfo >` (p. 2349), `decaf::lang::Pointer< activemq::threads::TaskRunner >` (p. 2349), `decaf::lang::Pointer< LockHandle >` (p. 2349), `decaf::lang::Pointer< ConsumerInfo >` (p. 2349), `decaf::lang::Pointer< ConnectionId >` (p. 2349), `decaf::lang::Pointer< decaf::lang::Runnable >` (p. 2349), `decaf::lang::Pointer< BackupTransportPool >` (p. 2349), `decaf::lang::Pointer< commands::BrokerInfo >` (p. 2349), `decaf::lang::Pointer< ProducerInfo >` (p. 2349), `decaf::lang::Pointer< decaf::lang::Thread >` (p. 2349), `decaf::lang::Pointer< MessageId >` (p. 2349),

decaf::lang::Pointer< Response > (p. 2349), decaf::lang::Pointer< SessionId > (p. 2349), decaf::lang::Pointer< cms::Destination > (p. 2349), decaf::lang::Pointer< TransportListener > (p. 2349), decaf::lang::Pointer< commands::TransactionId > (p. 2349), decaf::lang::Pointer< ActiveMQDestination > (p. 2349), decaf::lang::Pointer< ProducerId > (p. 2349), decaf::lang::Pointer< ResponseBuilder > (p. 2349), decaf::lang::Pointer< SessionInfo > (p. 2349), decaf::lang::Pointer< commands::Message > (p. 2349), decaf::lang::Pointer< transport::Transport > (p. 2349), decaf::lang::Pointer< InactivityMonitorData > (p. 2349), decaf::lang::Pointer< Tracked > (p. 2349), decaf::lang::Pointer< ConnectionInfo > (p. 2349), decaf::lang::Pointer< commands::MessageAck > (p. 2349), decaf::lang::Pointer< core::ActiveMQAckHandler > (p. 2349), decaf::lang::Pointer< Exception > (p. 2349), decaf::lang::Pointer< commands::ConsumerInfo > (p. 2349), decaf::lang::Pointer< ConsumerId > (p. 2349), decaf::lang::Pointer< URIPool > (p. 2349), decaf::lang::Pointer< decaf::util::Properties > (p. 2349), and decaf::lang::Pointer< TransactionId > (p. 2349).

References decaf::util::concurrent::atomic::AtomicInteger::decrementAndGet().

6.105.2.2 void decaf::lang::AtomicRefCounter::swap (AtomicRefCounter & *other*) [inline, protected]

Swaps this instance's reference counter with the one given, this allows for copy-and-swap semantics of this object.

Parameters

other The value to swap with this one's.

The documentation for this class was generated from the following file:

- src/main/decaf/lang/Pointer.h

6.106 decaf::util::concurrent::atomic::AtomicReference< T > Class Template Reference

An Pointer reference that may be updated atomically.

```
#include <src/main/decaf/util/concurrent/atomic/AtomicReference.h>
```

Public Member Functions

- **AtomicReference** ()
- **AtomicReference** (T *value)
- virtual ~**AtomicReference** ()
- T * **get** () const
Gets the Current Value.
- void **set** (T *newValue)
Sets the Current value of this Reference.
- bool **compareAndSet** (T *expect, T *update)

Atomically sets the value to the given updated value if the current value == the expected value.

- `T * getAndSet (T *newValue)`
Atomically sets to the given value and returns the old value.
- `std::string toString () const`
Returns the String representation of the current value.

6.106.1 Detailed Description

```
template<typename T> class decaf::util::concurrent::atomic::AtomicReference< T >
```

An Pointer reference that may be updated atomically.

6.106.2 Constructor & Destructor Documentation

6.106.2.1 `template<typename T >
decaf::util::concurrent::atomic::AtomicReference< T
>::AtomicReference () [inline]`

6.106.2.2 `template<typename T >
decaf::util::concurrent::atomic::AtomicReference< T
>::AtomicReference (T * value) [inline]`

6.106.2.3 `template<typename T > virtual
decaf::util::concurrent::atomic::AtomicReference< T
>::~~AtomicReference () [inline, virtual]`

6.106.3 Member Function Documentation

6.106.3.1 `template<typename T > bool
decaf::util::concurrent::atomic::AtomicReference< T
>::compareAndSet (T * expect, T * update) [inline]`

Atomically sets the value to the given updated value if the current value == the expected value.

Parameters

expect - the expected value
update - the new value

Returns

true if successful. False return indicates that the actual value was not equal to the expected value.

6.106.3.2 `template<typename T > T*
decaf::util::concurrent::atomic::AtomicReference< T >::get
() const [inline]`

Gets the Current Value.

Returns

the current value of this Reference.

6.106.3.3 `template<typename T > T*
decaf::util::concurrent::atomic::AtomicReference< T
>::getAndSet (T * new Value) [inline]`

Atomically sets to the given value and returns the old value.

Parameters

new Value- the new value

Returns

the previous value.

6.106.3.4 `template<typename T > void
decaf::util::concurrent::atomic::AtomicReference< T >::set
(T * new Value) [inline]`

Sets the Current value of this Reference.

Parameters

new Value The new Value of this Reference.

6.106.3.5 `template<typename T > std::string
decaf::util::concurrent::atomic::AtomicReference< T
>::toString () const [inline]`

Returns the String representation of the current value.

Returns

string representation of the current value.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/atomic/AtomicReference.h`

6.107 activemq::transport::failover::BackupTransport Class Reference

```
#include <src/main/activemq/transport/failover/BackupTransport.h>
```

Inheritance diagram for activemq::transport::failover::BackupTransport:

Public Member Functions

- **BackupTransport** (**BackupTransportPool** *failover)
- virtual **~BackupTransport** ()
- **decaf::net::URI** **getUri** () const
Gets the URI assigned to this Backup.
- void **setUri** (const **decaf::net::URI** &uri)
*Sets the URI assigned to this **Transport** (p. 3066).*
- const **Pointer**< **Transport** > & **getTransport** ()
Gets the currently held transport.
- void **setTransport** (const **Pointer**< **Transport** > &transport)
Sets the held transport, if not NULL then start to listen for exceptions from the held transport.
- virtual void **onException** (const **decaf::lang::Exception** &ex)
Event handler for an exception from a command transport.
- bool **isClosed** () const
*Has the **Transport** (p. 3066) been shutdown and no longer usable.*
- void **setClosed** (bool value)
*Sets the closed flag on this **Transport** (p. 3066).*

6.107.1 Constructor & Destructor Documentation

- 6.107.1.1 **activemq::transport::failover::BackupTransport::BackupTransport** (**BackupTransportPool** * *failover*)
- 6.107.1.2 virtual **activemq::transport::failover::BackupTransport::~~BackupTransport** ()
[virtual]

6.107.2 Member Function Documentation

- 6.107.2.1 const **Pointer**<**Transport**>& **activemq::transport::failover::BackupTransport::getTransport** () [inline]

Gets the currently held transport.

Returns

pointer to the held transport or NULL if not set.

6.107.2.2 `decaf::net::URI activemq::transport::failover::BackupTransport::getUri () const` [inline]

Gets the URI assigned to this Backup.

Returns

the assigned URI

6.107.2.3 `bool activemq::transport::failover::BackupTransport::isClosed () const` [inline]

Has the **Transport** (p. 3066) been shutdown and no longer usable.

Returns

true if the **Transport** (p. 3066)

6.107.2.4 `virtual void activemq::transport::failover::BackupTransport::onException (const decaf::lang::Exception & ex)` [virtual]

Event handler for an exception from a command transport.

The **BackupTransport** (p. 591) closes its internal **Transport** (p. 3066) when an exception is received and returns the URI to the pool of URIs to attempt connections to.

Parameters

ex The exception that was passed to this listener to handle.

Implements **activemq::transport::TransportListener** (p. 3082).

6.107.2.5 `void activemq::transport::failover::BackupTransport::setClosed (bool value)` [inline]

Sets the closed flag on this **Transport** (p. 3066).

Parameters

value - true for closed.

6.107.2.6 `void activemq::transport::failover::BackupTransport::setTransport (const Pointer< Transport > & transport)` [inline]

Sets the held transport, if not NULL then start to listen for exceptions from the held transport.

Parameters

transport The transport to hold.

6.107.2.7 `void activemq::transport::failover::BackupTransport::setUri (const decaf::net::URI & uri) [inline]`

Sets the URI assigned to this **Transport** (p. 3066).

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/failover/BackupTransport.h`

6.108 `activemq::transport::failover::BackupTransportPool` Class Reference

`#include <src/main/activemq/transport/failover/BackupTransportPool.h>`

Inheritance diagram for `activemq::transport::failover::BackupTransportPool`:

Public Member Functions

- **BackupTransportPool** (const **Pointer**< **CompositeTaskRunner** > &taskRunner, const **Pointer**< **CloseTransportsTask** > &closeTask, const **Pointer**< **URIPool** > &uriPool)
- **BackupTransportPool** (int backupPoolSize, const **Pointer**< **CompositeTaskRunner** > &taskRunner, const **Pointer**< **CloseTransportsTask** > &closeTask, const **Pointer**< **URIPool** > &uriPool)
- virtual **~BackupTransportPool** ()
- virtual bool **isPending** () const
Return true if we don't currently have enough Connected Transports.
- **Pointer**< **BackupTransport** > **getBackup** ()
*Get a Connected **Transport** (p. 3066) from the pool of Backups if any are present, otherwise it return a NULL Pointer.*
- virtual bool **iterate** ()
Connect to a Backup Broker if we haven't already connected to the max number of Backups.
- int **getBackupPoolSize** () const
Gets the Max number of Backups this Task will create.
- void **setBackupPoolSize** (int size)
Sets the Max number of Backups this Task will create.
- bool **isEnabled** () const
*Gets if the backup **Transport** (p. 3066) Pool has been enabled or not, when not enabled no backups are created and any that were are destroyed.*
- void **setEnabled** (bool value)
*Sets if this Backup **Transport** (p. 3066) Pool is enabled.*

Friends

- class **BackupTransport**

6.108.1 Constructor & Destructor Documentation

6.108.1.1 `activemq::transport::failover::BackupTransportPool::BackupTransportPool (const Pointer< CompositeTaskRunner > & taskRunner, const Pointer< CloseTransportsTask > & closeTask, const Pointer< URIPool > & uriPool)`

6.108.1.2 `activemq::transport::failover::BackupTransportPool::BackupTransportPool (int backupPoolSize, const Pointer< CompositeTaskRunner > & taskRunner, const Pointer< CloseTransportsTask > & closeTask, const Pointer< URIPool > & uriPool)`

6.108.1.3 `virtual
activemq::transport::failover::BackupTransportPool::~~BackupTransportPool
() [virtual]`

6.108.2 Member Function Documentation

6.108.2.1 `Pointer<BackupTransport> ac-
tivemq::transport::failover::BackupTransportPool::getBackup ()`

Get a Connected **Transport** (p. 3066) from the pool of Backups if any are present, otherwise it return a NULL Pointer.

Returns

Pointer to a Connected **Transport** (p. 3066) or NULL

6.108.2.2 `int ac-
tivemq::transport::failover::BackupTransportPool::getBackupPoolSize () const [inline]`

Gets the Max number of Backups this Task will create.

Returns

the max number of active BackupTransports that will be created.

6.108.2.3 `bool activemq::transport::failover::BackupTransportPool::isEnabled () const [inline]`

Gets if the backup **Transport** (p. 3066) Pool has been enabled or not, when not enabled no backups are created and any that were are destroyed.

Returns

true if enable.

6.108.2.4 `virtual bool activemq::transport::failover::BackupTransportPool::isPending () const` [virtual]

Return true if we don't currently have enough Connected Transports.

Implements `activemq::threads::CompositeTask` (p. 1016).

6.108.2.5 `virtual bool activemq::transport::failover::BackupTransportPool::iterate ()` [virtual]

Connect to a Backup Broker if we haven't already connected to the max number of Backups.

Implements `activemq::threads::Task` (p. 2956).

6.108.2.6 `void activemq::transport::failover::BackupTransportPool::setBackupPoolSize (int size)` [inline]

Sets the Max number of Backups this Task will create.

Parameters

size - the max number of active BackupTransports that will be created.

6.108.2.7 `void activemq::transport::failover::BackupTransportPool::setEnabled (bool value)`

Sets if this Backup **Transport** (p. 3066) Pool is enabled.

When not enabled no Backups are created and any that were are destroyed.

Parameters

value - true to enable backup creation, false to disable.

6.108.3 Friends And Related Function Documentation

6.108.3.1 `friend class BackupTransport` [friend]

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/failover/BackupTransportPool.h`

6.109 activemq::commands::BaseCommand Class Reference

```
#include <src/main/activemq/commands/BaseCommand.h>
```

Inheritance diagram for `activemq::commands::BaseCommand`:

Public Member Functions

- **BaseCommand** ()
- virtual **~BaseCommand** ()
- virtual void **setCommandId** (int id)
*Sets the **Command** (p. 991) Id of this **Message** (p. 2018).*
- virtual int **getCommandId** () const
*Gets the **Command** (p. 991) Id of this **Message** (p. 2018).*
- virtual void **setResponseRequired** (const bool required)
*Set if this **Message** (p. 2018) requires a **Response** (p. 2611).*
- virtual bool **isResponseRequired** () const
*Is a **Response** (p. 2611) required for this **Command** (p. 991).*
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1372) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent.*
- virtual bool **isConnectionInfo** () const
- virtual bool **isConsumerInfo** () const
- virtual bool **isBrokerInfo** () const
- virtual bool **isMessage** () const
- virtual bool **isMessageAck** () const
- virtual bool **isKeepAliveInfo** () const
- virtual bool **isMessageDispatch** () const
- virtual bool **isMessageDispatchNotification** () const
- virtual bool **isProducerAck** () const
- virtual bool **isProducerInfo** () const
- virtual bool **isResponse** () const
- virtual bool **isRemoveInfo** () const
- virtual bool **isRemoveSubscriptionInfo** () const
- virtual bool **isShutdownInfo** () const
- virtual bool **isTransactionInfo** () const
- virtual bool **isWireFormatInfo** () const

6.109.1 Constructor & Destructor Documentation

6.109.1.1 `activemq::commands::BaseCommand::BaseCommand ()` [inline]

6.109.1.2 `virtual activemq::commands::BaseCommand::~~BaseCommand ()`
[inline, virtual]

6.109.2 Member Function Documentation

6.109.2.1 `virtual void activemq::commands::BaseCommand::copyDataStructure (const DataStructure * src)` [inline, virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

`src` - Source Object

Implements `activemq::commands::DataStructure` (p. 1374).

Reimplemented in `activemq::commands::ActiveMQBlobMessage` (p. 143), `activemq::commands::ActiveMQBytesMessage` (p. 170), `activemq::commands::ActiveMQMapMessage` (p. 276), `activemq::commands::ActiveMQMessage` (p. 306), `activemq::commands::ActiveMQObjectMessage` (p. 346), `activemq::commands::ActiveMQStreamMessage` (p. 425), `activemq::commands::ActiveMQTextMessage` (p. 527), `activemq::commands::BrokerError` (p. 687), `activemq::commands::BrokerInfo` (p. 716), `activemq::commands::ConnectionControl` (p. 1057), `activemq::commands::ConnectionError` (p. 1081), `activemq::commands::ConnectionInfo` (p. 1130), `activemq::commands::ConsumerControl` (p. 1167), `activemq::commands::ConsumerInfo` (p. 1217), `activemq::commands::ControlCommand` (p. 1244), `activemq::commands::DataArrayResponse` (p. 1269), `activemq::commands::DataResponse` (p. 1308), `activemq::commands::DestinationInfo` (p. 1392), `activemq::commands::ExceptionResponse` (p. 1485), `activemq::commands::FlushCommand` (p. 1574), `activemq::commands::IntegerResponse` (p. 1668), `activemq::commands::KeepAliveInfo` (p. 1814), `activemq::commands::Message` (p. 2023), `activemq::commands::MessageAck` (p. 2057), `activemq::commands::MessageDispatch` (p. 2085), `activemq::commands::MessageDispatchNotification` (p. 2117), `activemq::commands::MessagePull` (p. 2204), `activemq::commands::ProducerAck` (p. 2422), `activemq::commands::ProducerInfo` (p. 2471), `activemq::commands::RemoveInfo` (p. 2538), `activemq::commands::RemoveSubscriptionInfo` (p. 2562), `activemq::commands::ReplayCommand` (p. 2585), `activemq::commands::Response` (p. 2612), `activemq::commands::SessionInfo` (p. 2703), `activemq::commands::ShutdownInfo` (p. 2758), `activemq::commands::TransactionInfo` (p. 3038), and `activemq::commands::WireFormatInfo` (p. 3156).

References `getCommandId()`, and `isResponseRequired()`.

6.109.2.2 `virtual bool activemq::commands::BaseCommand::equals (const DataStructure * value) const` [inline, virtual]

Compares the `DataStructure` (p. 1372) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Implements **activemq::commands::DataStructure** (p. 1374).

Reimplemented in **activemq::commands::ActiveMQBlobMessage** (p. 144), **activemq::commands::ActiveMQBytesMessage** (p. 171), **activemq::commands::ActiveMQMapMessage** (p. 276), **activemq::commands::ActiveMQMessage** (p. 306), **activemq::commands::ActiveMQMessageTemplate< T >** (p. 331), **activemq::commands::ActiveMQObjectMessage** (p. 346), **activemq::commands::ActiveMQStreamMessage** (p. 426), **activemq::commands::ActiveMQTextMessage** (p. 527), **activemq::commands::BrokerInfo** (p. 716), **activemq::commands::ConnectionControl** (p. 1057), **activemq::commands::ConnectionError** (p. 1081), **activemq::commands::ConnectionInfo** (p. 1131), **activemq::commands::ConsumerControl** (p. 1168), **activemq::commands::ConsumerInfo** (p. 1217), **activemq::commands::ControlCommand** (p. 1245), **activemq::commands::DataArrayResponse** (p. 1269), **activemq::commands::DataResponse** (p. 1309), **activemq::commands::DestinationInfo** (p. 1392), **activemq::commands::ExceptionResponse** (p. 1485), **activemq::commands::FlushCommand** (p. 1574), **activemq::commands::IntegerResponse** (p. 1668), **activemq::commands::KeepAliveInfo** (p. 1814), **activemq::commands::Message** (p. 2023), **activemq::commands::MessageAck** (p. 2057), **activemq::commands::MessageDispatch** (p. 2086), **activemq::commands::MessageDispatchNotification** (p. 2117), **activemq::commands::MessagePull** (p. 2204), **activemq::commands::ProducerAck** (p. 2422), **activemq::commands::ProducerInfo** (p. 2471), **activemq::commands::RemoveInfo** (p. 2538), **activemq::commands::RemoveSubscriptionInfo** (p. 2562), **activemq::commands::ReplayCommand** (p. 2586), **activemq::commands::Response** (p. 2613), **activemq::commands::SessionInfo** (p. 2703), **activemq::commands::ShutdownInfo** (p. 2758), **activemq::commands::TransactionInfo** (p. 3039), **activemq::commands::WireFormatInfo** (p. 3156), **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 331), **activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >** (p. 331), **activemq::commands::ActiveMQMessageTemplate< cms::Message >** (p. 331), **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 331), **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 331), and **activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >** (p. 331).

References **activemq::commands::BaseDataStructure::equals()**.

6.109.2.3 virtual int activemq::commands::BaseCommand::getCommandId ()
const [inline, virtual]

Gets the **Command** (p. 991) Id of this **Message** (p. 2018).

Returns

Command (p. 991) Id

Implements **activemq::commands::Command** (p. 992).

Referenced by **copyDataStructure()**.

6.109.2.4 `virtual bool activemq::commands::BaseCommand::isBrokerInfo ()`
`const [inline, virtual]`

Implements `activemq::commands::Command` (p. 992).

Reimplemented in `activemq::commands::BrokerInfo` (p. 718).

6.109.2.5 `virtual bool activemq::commands::BaseCommand::isConnectionInfo ()`
`const [inline, virtual]`

Implements `activemq::commands::Command` (p. 992).

Reimplemented in `activemq::commands::ConnectionInfo` (p. 1132).

6.109.2.6 `virtual bool activemq::commands::BaseCommand::isConsumerInfo ()`
`const [inline, virtual]`

Implements `activemq::commands::Command` (p. 992).

Reimplemented in `activemq::commands::ConsumerInfo` (p. 1219).

6.109.2.7 `virtual bool activemq::commands::BaseCommand::isKeepAliveInfo ()`
`const [inline, virtual]`

Implements `activemq::commands::Command` (p. 992).

Reimplemented in `activemq::commands::KeepAliveInfo` (p. 1814).

6.109.2.8 `virtual bool activemq::commands::BaseCommand::isMessage () const`
`[inline, virtual]`

Implements `activemq::commands::Command` (p. 993).

Reimplemented in `activemq::commands::Message` (p. 2030).

6.109.2.9 `virtual bool activemq::commands::BaseCommand::isMessageAck ()`
`const [inline, virtual]`

Implements `activemq::commands::Command` (p. 993).

Reimplemented in `activemq::commands::MessageAck` (p. 2058).

6.109.2.10 `virtual bool activemq::commands::BaseCommand::isMessageDispatch () const`
`[inline, virtual]`

Implements `activemq::commands::Command` (p. 993).

Reimplemented in `activemq::commands::MessageDispatch` (p. 2087).

6.109.2.11 `virtual bool activemq::commands::BaseCommand::isMessageDispatchNotification () const [inline, virtual]`

Implements `activemq::commands::Command` (p. 993).

Reimplemented in `activemq::commands::MessageDispatchNotification` (p. 2118).

6.109.2.12 `virtual bool activemq::commands::BaseCommand::isProducerAck () const [inline, virtual]`

Implements `activemq::commands::Command` (p. 993).

Reimplemented in `activemq::commands::ProducerAck` (p. 2422).

6.109.2.13 `virtual bool activemq::commands::BaseCommand::isProducerInfo () const [inline, virtual]`

Implements `activemq::commands::Command` (p. 993).

Reimplemented in `activemq::commands::ProducerInfo` (p. 2472).

6.109.2.14 `virtual bool activemq::commands::BaseCommand::isRemoveInfo () const [inline, virtual]`

Implements `activemq::commands::Command` (p. 993).

Reimplemented in `activemq::commands::RemoveInfo` (p. 2539).

6.109.2.15 `virtual bool activemq::commands::BaseCommand::isRemoveSubscriptionInfo () const [inline, virtual]`

Implements `activemq::commands::Command` (p. 993).

Reimplemented in `activemq::commands::RemoveSubscriptionInfo` (p. 2563).

6.109.2.16 `virtual bool activemq::commands::BaseCommand::isResponse () const [inline, virtual]`

Implements `activemq::commands::Command` (p. 994).

Reimplemented in `activemq::commands::Response` (p. 2613).

6.109.2.17 `virtual bool activemq::commands::BaseCommand::isResponseRequired () const [inline, virtual]`

Is a **Response** (p. 2611) required for this **Command** (p. 991).

Returns

true if a response is required.

Implements **activemq::commands::Command** (p. 994).

Referenced by `copyDataStructure()`.

6.109.2.18 `virtual bool activemq::commands::BaseCommand::isShutdownInfo ()
const [inline, virtual]`

Implements **activemq::commands::Command** (p. 994).

Reimplemented in **activemq::commands::ShutdownInfo** (p. 2759).

6.109.2.19 `virtual bool activemq::commands::BaseCommand::isTransactionInfo ()
const [inline, virtual]`

Implements **activemq::commands::Command** (p. 994).

Reimplemented in **activemq::commands::TransactionInfo** (p. 3039).

6.109.2.20 `virtual bool activemq::commands::BaseCommand::isWireFormatInfo ()
const [inline, virtual]`

Implements **activemq::commands::Command** (p. 994).

Reimplemented in **activemq::commands::WireFormatInfo** (p. 3159).

6.109.2.21 `virtual void activemq::commands::BaseCommand::setCommandId (int
id) [inline, virtual]`

Sets the **Command** (p. 991) Id of this **Message** (p. 2018).

Parameters

id **Command** (p. 991) Id

Implements **activemq::commands::Command** (p. 994).

6.109.2.22 `virtual void activemq::commands::BaseCommand::setResponseRequired (const bool required) [inline, virtual]`

Set if this **Message** (p. 2018) requires a **Response** (p. 2611).

Parameters

required true if response is required

Implements **activemq::commands::Command** (p. 994).

6.109.2.23 `virtual std::string activemq::commands::BaseCommand::toString ()
const [inline, virtual]`

Returns a string containing the information for this **DataStructure** (p. 1372) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Implements `activemq::commands::Command` (p. 995).

Reimplemented in `activemq::commands::ActiveMQBlobMessage` (p. 146), `activemq::commands::ActiveMQBytesMessage` (p. 177), `activemq::commands::ActiveMQMapMessage` (p. 284), `activemq::commands::ActiveMQMessage` (p. 307), `activemq::commands::ActiveMQObjectMessage` (p. 346), `activemq::commands::ActiveMQStreamMessage` (p. 432), `activemq::commands::ActiveMQTextMessage` (p. 529), `activemq::commands::BrokerInfo` (p. 719), `activemq::commands::ConnectionControl` (p. 1058), `activemq::commands::ConnectionError` (p. 1082), `activemq::commands::ConnectionInfo` (p. 1133), `activemq::commands::ConsumerControl` (p. 1169), `activemq::commands::ConsumerInfo` (p. 1220), `activemq::commands::ControlCommand` (p. 1245), `activemq::commands::DataArrayResponse` (p. 1270), `activemq::commands::DataResponse` (p. 1309), `activemq::commands::DestinationInfo` (p. 1394), `activemq::commands::ExceptionResponse` (p. 1486), `activemq::commands::FlushCommand` (p. 1575), `activemq::commands::IntegerResponse` (p. 1669), `activemq::commands::KeepAliveInfo` (p. 1815), `activemq::commands::Message` (p. 2033), `activemq::commands::MessageAck` (p. 2059), `activemq::commands::MessageDispatch` (p. 2087), `activemq::commands::MessageDispatchNotification` (p. 2119), `activemq::commands::MessagePull` (p. 2206), `activemq::commands::ProducerAck` (p. 2423), `activemq::commands::ProducerInfo` (p. 2473), `activemq::commands::RemoveInfo` (p. 2539), `activemq::commands::RemoveSubscriptionInfo` (p. 2563), `activemq::commands::ReplayCommand` (p. 2586), `activemq::commands::Response` (p. 2614), `activemq::commands::SessionInfo` (p. 2704), `activemq::commands::ShutdownInfo` (p. 2759), `activemq::commands::TransactionInfo` (p. 3040), and `activemq::commands::WireFormatInfo` (p. 3162).

References `activemq::commands::BaseDataStructure::toString()`.

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/BaseCommand.h`

6.110 `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` Class Reference

Marshaling code for Open Wire Format for `BaseCommandMarshaller` (p. 603).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller`:

Public Member Functions

- `BaseCommandMarshaller ()`
- `virtual ~BaseCommandMarshaller ()`

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

6.110.1 Detailed Description

Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 603). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.110.2 Constructor & Destructor Documentation

6.110.2.1 **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller::BaseCommandMarshaller** () [inline]

6.110.2.2 **virtual activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller::~~BaseCommandMarshaller** () [inline, virtual]

6.110.3 Member Function Documentation

6.110.3.1 **virtual void activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller::looseMarshal** (**OpenWireFormat** * *wireFormat*, **commands::DataStructure** * *dataStructure*, **decaf::io::DataOutputStream** * *dataOut*) throw (**decaf::io::IOException**) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1342).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller` (p. 148), `activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller` (p. 184), `activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller` (p. 286), `activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller` (p. 309), `activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller` (p. 349), `activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller` (p. 437), `activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller` (p. 531), `activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller` (p. 723), `activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller` (p. 1061), `activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller` (p. 1085), `activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller` (p. 1136), `activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller` (p. 1172), `activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller` (p. 1224), `activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller` (p. 1248), `activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller` (p. 1272), `activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller` (p. 1312), `activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller` (p. 1401), `activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller` (p. 1496), `activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller` (p. 1585), `activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller` (p. 1679), `activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller` (p. 1825), `activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller` (p. 2070), `activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller` (p. 2105), `activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller` (p. 2130), `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2176), `activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller` (p. 2213), `activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller` (p. 2429), `activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller` (p. 2480), `activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller` (p. 2554), `activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller` (p. 2578), `activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller` (p. 2593), `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 2626), `activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller` (p. 2723), `activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller` (p. 2769), and `activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller` (p. 3046).

6.110.3.2 virtual void `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller::looseUnmarshal` (`OpenWireFormat * wireFormat`, `commands::DataStructure * dataStructure`, `decaf::io::DataInputStream * dataIn`) throw (`decaf::io::IOException`) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1348).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller` (p. 149), `activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller` (p. 184), `activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller` (p. 287), `activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller` (p. 309), `activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller` (p. 349), `activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller` (p. 438), `activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller` (p. 532), `activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller` (p. 724), `activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller` (p. 1062), `activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller` (p. 1085), `activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller` (p. 1137), `activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller` (p. 1173), `activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller` (p. 1225), `activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller` (p. 1248), `activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller` (p. 1273), `activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller` (p. 1312), `activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller` (p. 1401), `activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller` (p. 1497), `activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller` (p. 1586), `activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller` (p. 1680), `activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller` (p. 1826), `activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller` (p. 2070), `activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller` (p. 2106), `activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller` (p. 2130), `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2176), `activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller` (p. 2213), `activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller` (p. 2430), `activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller` (p. 2480), `activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller` (p. 2554), `activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller` (p. 2578), `activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller` (p. 2593), `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 2626), `activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller` (p. 2723), `activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller` (p. 2770), and `activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller` (p. 3047).

```
6.110.3.3 virtual int ac-
    tivemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller::tightMarshal1
    ( OpenWireFormat * wireFormat, commands::DataStructure
    * dataStructure, utils::BooleanStream * bs ) throw (
    decaf::io::IOException ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1354).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller` (p. 149), `activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller` (p. 184), `activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller` (p. 287), `activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller` (p. 309), `activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller` (p. 349), `activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller` (p. 438), `activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller` (p. 532), `activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller` (p. 724), `activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller` (p. 1062), `activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller` (p. 1086), `activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller` (p. 1137), `activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller` (p. 1173), `activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller` (p. 1225), `activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller` (p. 1249), `activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller` (p. 1273), `activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller` (p. 1312), `activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller` (p. 1402), `activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller` (p. 1497), `activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller` (p. 1586), `activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller` (p. 1680), `activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller` (p. 1826), `activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller` (p. 2070), `activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller` (p. 2106), `activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller` (p. 2130), `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2177), `activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller` (p. 2213), `activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller` (p. 2430), `activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller` (p. 2480), `activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller` (p. 2554), `activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller` (p. 2578), `activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller` (p. 2593), `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 2627), `activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller` (p. 2723), `activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller` (p. 2770), and `activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller` (p. 3047).

```

6.110.3.4 virtual void ac-
tivismq::wireformat::openwire::marshal::v3::BaseCommandMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1360).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller` (p. 149), `activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller` (p. 185), `activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller` (p. 287), `activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller` (p. 310), `activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller` (p. 350), `activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller` (p. 439), `activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller` (p. 532), `activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller` (p. 724), `activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller` (p. 1062), `activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller` (p. 1086), `activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller` (p. 1137), `activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller` (p. 1173), `activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller` (p. 1226), `activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller` (p. 1249), `activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller` (p. 1274), `activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller` (p. 1313), `activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller` (p. 1402), `activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller` (p. 1498), `activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller` (p. 1586), `activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller` (p. 1681), `activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller` (p. 1826), `activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller` (p. 2071), `activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller` (p. 2106), `activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller` (p. 2131), `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` (p. 2178), `activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller` (p. 2214), `activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller` (p. 2430), `activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller` (p. 2481), `activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller` (p. 2555), `activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller` (p. 2579), `activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller` (p. 2594), `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 2627),

activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller (p. 2724), activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller (p. 2770), and activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller (p. 3047).

6.110.3.5 virtual void activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Implements activemq::wireformat::openwire::marshal::DataStreamMarshaller (p. 1366).

Reimplemented in activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller (p. 150), activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller (p. 185), activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller (p. 288), activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller (p. 310), activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller (p. 350), activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller (p. 439), activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller (p. 533), activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller (p. 725), activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller (p. 1063), activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller (p. 1087), activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller (p. 1138), activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller (p. 1174), activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller (p. 1226), activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller (p. 1250), activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller (p. 1274), activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller (p. 1313), activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller (p. 1402), activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller (p. 1498), activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller (p. 1587), activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller (p. 1681), activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller (p. 1827), activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller (p. 2071), activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller (p. 2107), activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller (p. 2131), activemq::wireformat::openwire::marshal::v3::MessageMarshaller (p. 2178), activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller (p. 2214), activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller (p. 2431), activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller (p. 2481),

activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller (p. 2555), **activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller** (p. 2579), **activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller** (p. 2594), **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 2628), **activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller** (p. 2724), **activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller** (p. 2771), and **activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller** (p. 3048).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h`

6.111 **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** Class Reference

Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 610).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller**:

Public Member Functions

- **BaseCommandMarshaller** ()
- virtual **~BaseCommandMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.111.1 Detailed Description

Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 610). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.111.2 Constructor & Destructor Documentation

6.111.2.1 `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller::BaseCommandMarshaller ()` [inline]

6.111.2.2 `virtual activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller::~~BaseCommandMarshaller ()` [inline, virtual]

6.111.3 Member Function Documentation

6.111.3.1 `virtual void activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryWriter that provides that data sink

Exceptions

- IOException* if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1342).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller` (p. 152), `activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller` (p. 188), `activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller` (p. 290), `activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller` (p. 313), `activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller` (p. 353), `activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller` (p. 441), `activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller` (p. 535), `activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller` (p. 727), `activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller` (p. 1065), `activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller` (p. 1089), `activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller` (p. 1144), `activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller` (p. 1180), `activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller` (p. 1228), `activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller` (p. 1256), `activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller` (p. 1280), `activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller` (p. 1316),

activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller (p. 1409),
 activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller (p. 1492),
 activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller (p. 1581),
 activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller (p. 1675),
 activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller (p. 1833),
 activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller (p. 2078),
 activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller (p. 2109),
 activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller
 (p. 2134), activemq::wireformat::openwire::marshal::v1::MessageMarshaller (p. 2184),
 activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller (p. 2217),
 activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller (p. 2441),
 activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller (p. 2492),
 activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller (p. 2542),
 activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller
 (p. 2574), activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller
 (p. 2605), activemq::wireformat::openwire::marshal::v1::ResponseMarshaller (p. 2635),
 activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller (p. 2711),
 activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller (p. 2761), and
 activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller (p. 3050).

6.111.3.2 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller::looseUnmarshal
(OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn) throw (
decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1348).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller**
 (p. 153), **activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller**
 (p. 188), **activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller**
 (p. 291), **activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller**
 (p. 313), **activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller**
 (p. 353), **activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller**
 (p. 442), **activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller**
 (p. 536), **activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller** (p. 728),
activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller (p. 1066),
activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller (p. 1089),
activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller (p. 1145),
activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller (p. 1181),
activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller (p. 1229),
activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller (p. 1256),
activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller (p. 1281),

activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller (p. 1316),
 activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller (p. 1409),
 activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller (p. 1493),
 activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller (p. 1582),
 activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller (p. 1676),
 activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller (p. 1834),
 activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller (p. 2078),
 activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller (p. 2110),
 activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller
 (p. 2134), activemq::wireformat::openwire::marshal::v1::MessageMarshaller (p. 2184),
 activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller (p. 2217),
 activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller (p. 2442),
 activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller (p. 2492),
 activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller (p. 2542),
 activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller
 (p. 2574), activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller
 (p. 2605), activemq::wireformat::openwire::marshal::v1::ResponseMarshaller (p. 2635),
 activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller (p. 2711),
 activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller (p. 2762), and
 activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller (p. 3051).

6.111.3.3 virtual int `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller::tightMarshal1`
 (`OpenWireFormat * wireFormat`, `commands::DataStructure`
 * `dataStructure`, `utils::BooleanStream * bs`) throw (`decaf::io::IOException`) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1354).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller`
 (p. 153), `activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller`
 (p. 188), `activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller`
 (p. 291), `activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller`
 (p. 313), `activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller`
 (p. 353), `activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller`
 (p. 442), `activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller`
 (p. 536), `activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller` (p. 728),
`activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller` (p. 1066),

activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller (p. 1090),
 activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller (p. 1145),
 activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller (p. 1181),
 activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller (p. 1229),
 activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller (p. 1256),
 activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller (p. 1281),
 activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller (p. 1316),
 activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller (p. 1409),
 activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller (p. 1493),
 activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller (p. 1582),
 activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller (p. 1676),
 activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller (p. 1834),
 activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller (p. 2078),
 activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller (p. 2110),
 activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller
 (p. 2134), activemq::wireformat::openwire::marshal::v1::MessageMarshaller (p. 2185),
 activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller (p. 2217),
 activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller (p. 2442),
 activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller (p. 2492),
 activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller (p. 2543),
 activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller
 (p. 2574), activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller
 (p. 2605), activemq::wireformat::openwire::marshal::v1::ResponseMarshaller (p. 2636),
 activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller (p. 2711),
 activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller (p. 2762), and
 activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller (p. 3051).

6.111.3.4 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller::tightMarshal2
(OpenWireFormat * *wireFormat*, commands::DataStructure
*** *dataStructure*, decaf::io::DataOutputStream * *dataOut*,**
utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1360).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller**
 (p. 153), **activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller**
 (p. 189), **activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller**
 (p. 291), **activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller**
 (p. 314), **activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller**

(p. 354), `activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller` (p. 442), `activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller` (p. 536), `activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller` (p. 728), `activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller` (p. 1066), `activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller` (p. 1090), `activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller` (p. 1145), `activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller` (p. 1181), `activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller` (p. 1229), `activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller` (p. 1257), `activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller` (p. 1282), `activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller` (p. 1317), `activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller` (p. 1410), `activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller` (p. 1494), `activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller` (p. 1582), `activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller` (p. 1677), `activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller` (p. 1834), `activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller` (p. 2079), `activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller` (p. 2110), `activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller` (p. 2135), `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2186), `activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller` (p. 2218), `activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller` (p. 2442), `activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller` (p. 2493), `activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller` (p. 2543), `activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller` (p. 2575), `activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller` (p. 2606), `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 2636), `activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller` (p. 2712), `activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller` (p. 2762), and `activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller` (p. 3051).

6.111.3.5 `virtual void activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1366).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller` (p. 154), `activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller`

(p. 189), `activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller` (p. 292), `activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller` (p. 314), `activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller` (p. 354), `activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller` (p. 443), `activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller` (p. 537), `activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller` (p. 729), `activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller` (p. 1067), `activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller` (p. 1090), `activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller` (p. 1146), `activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller` (p. 1182), `activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller` (p. 1230), `activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller` (p. 1257), `activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller` (p. 1282), `activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller` (p. 1317), `activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller` (p. 1410), `activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller` (p. 1494), `activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller` (p. 1583), `activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller` (p. 1677), `activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller` (p. 1835), `activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller` (p. 2079), `activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller` (p. 2111), `activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller` (p. 2135), `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` (p. 2186), `activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller` (p. 2218), `activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller` (p. 2443), `activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller` (p. 2493), `activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller` (p. 2543), `activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller` (p. 2575), `activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller` (p. 2606), `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 2637), `activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller` (p. 2712), `activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller` (p. 2763), and `activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller` (p. 3052).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h`

6.112 `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` Class Reference

Marshaling code for Open Wire Format for `BaseCommandMarshaller` (p. 616).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller`:

Public Member Functions

- `BaseCommandMarshaller ()`
- `virtual ~BaseCommandMarshaller ()`

- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.112.1 Detailed Description

Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p.616). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.112.2 Constructor & Destructor Documentation

6.112.2.1 activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller::BaseCommandMarshaller () [inline]

6.112.2.2 virtual activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller::~~BaseCommandMarshaller () [inline, virtual]

6.112.3 Member Function Documentation

6.112.3.1 virtual void activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1342).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller` (p. 156), `activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller` (p. 192), `activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller` (p. 294), `activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller` (p. 317), `activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller` (p. 357), `activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller` (p. 445), `activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller` (p. 539), `activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller` (p. 731), `activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller` (p. 1069), `activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller` (p. 1093), `activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller` (p. 1140), `activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller` (p. 1176), `activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller` (p. 1232), `activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller` (p. 1252), `activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller` (p. 1276), `activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller` (p. 1324), `activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller` (p. 1405), `activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller` (p. 1500), `activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller` (p. 1589), `activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller` (p. 1683), `activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller` (p. 1829), `activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller` (p. 2074), `activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller` (p. 2101), `activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller` (p. 2126), `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2172), `activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller` (p. 2225), `activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller` (p. 2433), `activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller` (p. 2488), `activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller` (p. 2558), `activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller` (p. 2582), `activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller` (p. 2601), `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 2639), `activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller` (p. 2715), `activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller` (p. 2765), and `activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller` (p. 3054).

6.112.3.2 virtual void `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller::looseUnmarshal` (`OpenWireFormat * wireFormat`, `commands::DataStructure * dataStructure`, `decaf::io::DataInputStream * dataIn`) throw (`decaf::io::IOException`) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1348).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller` (p. 157), `activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller` (p. 192), `activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller` (p. 295), `activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller` (p. 317), `activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller` (p. 357), `activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller` (p. 446), `activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller` (p. 540), `activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller` (p. 732), `activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller` (p. 1070), `activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller` (p. 1093), `activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller` (p. 1141), `activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller` (p. 1177), `activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller` (p. 1233), `activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller` (p. 1252), `activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller` (p. 1277), `activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller` (p. 1324), `activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller` (p. 1405), `activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller` (p. 1501), `activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller` (p. 1590), `activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller` (p. 1684), `activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller` (p. 1830), `activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller` (p. 2074), `activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller` (p. 2102), `activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller` (p. 2126), `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2172), `activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller` (p. 2225), `activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller` (p. 2434), `activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller` (p. 2488), `activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller` (p. 2558), `activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller` (p. 2582), `activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller` (p. 2601), `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 2640), `activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller` (p. 2715), `activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller` (p. 2766), and `activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller` (p. 3055).

6.112.3.3 virtual int `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller::tightMarshal1`
(`OpenWireFormat * wireFormat`, `commands::DataStructure * dataStructure`, `utils::BooleanStream * bs`) throw (`decaf::io::IOException`) [virtual]

Write the booleans that this object uses to a `BooleanStream`.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1354).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller` (p. 157), `activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller` (p. 192), `activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller` (p. 295), `activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller` (p. 317), `activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller` (p. 357), `activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller` (p. 446), `activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller` (p. 540), `activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller` (p. 732), `activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller` (p. 1070), `activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller` (p. 1093), `activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller` (p. 1141), `activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller` (p. 1177), `activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller` (p. 1233), `activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller` (p. 1253), `activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller` (p. 1277), `activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller` (p. 1324), `activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller` (p. 1405), `activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller` (p. 1501), `activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller` (p. 1590), `activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller` (p. 1684), `activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller` (p. 1830), `activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller` (p. 2074), `activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller` (p. 2102), `activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller` (p. 2126), `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2173), `activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller` (p. 2225), `activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller` (p. 2434), `activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller` (p. 2488), `activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller` (p. 2558), `activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller` (p. 2582), `activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller` (p. 2601), `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 2640), `activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller` (p. 2715), `activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller` (p. 2766), and `activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller` (p. 3055).

6.112.3.4 virtual void activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1360).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller` (p. 157), `activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller` (p. 193), `activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller` (p. 295), `activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller` (p. 318), `activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller` (p. 358), `activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller` (p. 446), `activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller` (p. 540), `activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller` (p. 732), `activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller` (p. 1070), `activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller` (p. 1094), `activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller` (p. 1141), `activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller` (p. 1177), `activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller` (p. 1233), `activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller` (p. 1253), `activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller` (p. 1278), `activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller` (p. 1325), `activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller` (p. 1406), `activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller` (p. 1502), `activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller` (p. 1590), `activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller` (p. 1685), `activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller` (p. 1830), `activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller` (p. 2075), `activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller` (p. 2102), `activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller` (p. 2127), `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2174), `activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller` (p. 2226), `activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller` (p. 2434), `activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller` (p. 2489), `activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller` (p. 2559), `activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller` (p. 2583), `activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller` (p. 2602), `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 2641),

`activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller` (p. 2716), `activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller` (p. 2766), and `activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller` (p. 3055).

6.112.3.5 virtual void `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller::tightUnmarshal` (`OpenWireFormat * wireFormat`, `commands::DataStructure * dataStructure`, `decaf::io::DataInputStream * dataIn`, `utils::BooleanStream * bs`) throw (`decaf::io::IOException`) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1366).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller` (p. 158), `activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller` (p. 193), `activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller` (p. 296), `activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller` (p. 318), `activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller` (p. 358), `activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller` (p. 447), `activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller` (p. 541), `activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller` (p. 733), `activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller` (p. 1071), `activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller` (p. 1094), `activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller` (p. 1142), `activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller` (p. 1178), `activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller` (p. 1234), `activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller` (p. 1253), `activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller` (p. 1278), `activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller` (p. 1325), `activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller` (p. 1406), `activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller` (p. 1502), `activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller` (p. 1591), `activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller` (p. 1685), `activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller` (p. 1831), `activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller` (p. 2075), `activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller` (p. 2103), `activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller` (p. 2127), `activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2174), `activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller` (p. 2226), `activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller` (p. 2435), `activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller` (p. 2489),

activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller (p. 2559), **activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller** (p. 2583), **activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller** (p. 2602), **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 2641), **activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller** (p. 2716), **activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller** (p. 2767), and **activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller** (p. 3056).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h`

6.113 activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 623).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller`:

Public Member Functions

- **BaseCommandMarshaller** ()
- virtual **~BaseCommandMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.113.1 Detailed Description

Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 623). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.113.2 Constructor & Destructor Documentation

6.113.2.1 `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller::BaseCommandMarshaller ()` [inline]

6.113.2.2 `virtual activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller::~~BaseCommandMarshaller ()` [inline, virtual]

6.113.3 Member Function Documentation

6.113.3.1 `virtual void activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1342).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller` (p. 160), `activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller` (p. 196), `activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller` (p. 298), `activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller` (p. 321), `activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller` (p. 361), `activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller` (p. 449), `activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller` (p. 543), `activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller` (p. 735), `activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller` (p. 1073), `activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller` (p. 1097), `activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller` (p. 1148), `activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller` (p. 1184), `activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller` (p. 1236), `activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller` (p. 1260), `activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller` (p. 1284), `activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller` (p. 1320),

activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller (p. 1413),
 activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller (p. 1504),
 activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller (p. 1593),
 activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller (p. 1687),
 activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller (p. 1821),
 activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller (p. 2066),
 activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller (p. 2113),
 activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller
 (p. 2138),
 activemq::wireformat::openwire::marshal::v5::MessageMarshaller (p. 2180),
 activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller (p. 2221),
 activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller (p. 2437),
 activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller (p. 2484),
 activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller (p. 2546),
 activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller
 (p. 2566),
 activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller
 (p. 2597),
 activemq::wireformat::openwire::marshal::v5::ResponseMarshaller (p. 2630),
 activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller (p. 2719),
 activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller (p. 2777),
 and
 activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller (p. 3042).

6.113.3.2 virtual void `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller::looseUnmarshal`
 (`OpenWireFormat * wireFormat`, `commands::DataStructure * dataStructure`,
`decaf::io::DataInputStream * dataIn`) throw (`decaf::io::IOException`) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1348).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller`
 (p. 161), `activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller`
 (p. 196), `activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller`
 (p. 299), `activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller`
 (p. 321), `activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller`
 (p. 361), `activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller`
 (p. 450), `activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller`
 (p. 544), `activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller` (p. 736),
`activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller` (p. 1074),
`activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller` (p. 1097),
`activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller` (p. 1149),
`activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller` (p. 1185),
`activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller` (p. 1237),
`activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller` (p. 1260),
`activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller` (p. 1285),

activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller (p. 1320),
activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller (p. 1413), **activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller** (p. 1505),
activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller (p. 1594),
activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller (p. 1688),
activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller (p. 1822),
activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller (p. 2066), **activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller** (p. 2114), **activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller** (p. 2138), **activemq::wireformat::openwire::marshal::v5::MessageMarshaller** (p. 2180),
activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller (p. 2221),
activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller (p. 2438),
activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller (p. 2484),
activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller (p. 2546), **activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller** (p. 2566), **activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller** (p. 2597), **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 2631),
activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller (p. 2719), **activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller** (p. 2778), and **activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller** (p. 3043).

6.113.3.3 virtual int activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller::tightMarshal1
 (**OpenWireFormat** * *wireFormat*, **commands::DataStructure**
 * *dataStructure*, **utils::BooleanStream** * *bs*) throw (**decaf::io::IOException**) [virtual]

Write the booleans that this object uses to a **BooleanStream**.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - **BooleanStream** stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1354).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller** (p. 161), **activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller** (p. 196), **activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller** (p. 299), **activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller** (p. 321), **activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller** (p. 361), **activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller** (p. 450), **activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller** (p. 544), **activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller** (p. 736), **activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller** (p. 1074),

activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller (p. 1097),
 activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller (p. 1149),
 activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller (p. 1185),
 activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller (p. 1237),
 activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller (p. 1260),
 activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller (p. 1285),
 activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller (p. 1320),
 activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller (p. 1413),
 activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller (p. 1505),
 activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller (p. 1594),
 activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller (p. 1688),
 activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller (p. 1822),
 activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller (p. 2066),
 activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller (p. 2114),
 activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller
 (p. 2138),
 activemq::wireformat::openwire::marshal::v5::MessageMarshaller (p. 2181),
 activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller (p. 2221),
 activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller (p. 2438),
 activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller (p. 2484),
 activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller (p. 2546),
 activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller
 (p. 2567),
 activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller
 (p. 2597),
 activemq::wireformat::openwire::marshal::v5::ResponseMarshaller (p. 2631),
 activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller (p. 2719),
 activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller (p. 2778), and
 activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller (p. 3043).

6.113.3.4 virtual void `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller::tightMarshal2`
 (`OpenWireFormat * wireFormat`, `commands::DataStructure`
 * `dataStructure`, `decaf::io::DataOutputStream * dataOut`,
`utils::BooleanStream * bs`) throw (`decaf::io::IOException`) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1360).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller`
 (p. 161), `activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller`
 (p. 197), `activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller`
 (p. 299), `activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller`
 (p. 322), `activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller`

(p. 362), `activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller` (p. 450), `activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller` (p. 544), `activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller` (p. 736), `activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller` (p. 1074), `activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller` (p. 1098), `activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller` (p. 1149), `activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller` (p. 1185), `activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller` (p. 1237), `activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller` (p. 1261), `activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller` (p. 1286), `activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller` (p. 1321), `activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller` (p. 1414), `activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller` (p. 1506), `activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller` (p. 1594), `activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller` (p. 1689), `activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller` (p. 1822), `activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller` (p. 2067), `activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller` (p. 2114), `activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller` (p. 2139), `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2182), `activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller` (p. 2222), `activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller` (p. 2438), `activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller` (p. 2485), `activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller` (p. 2547), `activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller` (p. 2567), `activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller` (p. 2598), `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 2632), `activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller` (p. 2720), `activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller` (p. 2778), and `activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller` (p. 3044).

6.113.3.5 `virtual void activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1366).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller` (p. 162), `activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller`

(p. 197), [activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller](#) (p. 300), [activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller](#) (p. 322), [activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller](#) (p. 362), [activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller](#) (p. 451), [activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller](#) (p. 545), [activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller](#) (p. 737), [activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller](#) (p. 1075), [activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller](#) (p. 1098), [activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller](#) (p. 1150), [activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller](#) (p. 1186), [activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller](#) (p. 1238), [activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller](#) (p. 1261), [activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller](#) (p. 1286), [activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller](#) (p. 1321), [activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller](#) (p. 1414), [activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller](#) (p. 1506), [activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller](#) (p. 1595), [activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller](#) (p. 1689), [activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller](#) (p. 1823), [activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller](#) (p. 2067), [activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller](#) (p. 2115), [activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller](#) (p. 2139), [activemq::wireformat::openwire::marshal::v5::MessageMarshaller](#) (p. 2182), [activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller](#) (p. 2222), [activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller](#) (p. 2439), [activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller](#) (p. 2485), [activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller](#) (p. 2547), [activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller](#) (p. 2567), [activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller](#) (p. 2598), [activemq::wireformat::openwire::marshal::v5::ResponseMarshaller](#) (p. 2632), [activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller](#) (p. 2720), [activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller](#) (p. 2779), and [activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller](#) (p. 3044).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h`

6.114 activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 630).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller`:

Public Member Functions

- `BaseCommandMarshaller ()`
- `virtual ~BaseCommandMarshaller ()`

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

6.114.1 Detailed Description

Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 630). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.114.2 Constructor & Destructor Documentation

6.114.2.1 **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller::BaseCommandMarshaller**
() [inline]

6.114.2.2 **virtual**
activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller::~~BaseCommandMarshaller
() [inline, virtual]

6.114.3 Member Function Documentation

6.114.3.1 **virtual void** **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller::looseMarshal**
(**OpenWireFormat** * *wireFormat*, **commands::DataStructure** * *dataStructure*, **decaf::io::DataOutputStream** * *dataOut*) throw (**decaf::io::IOException**) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1342).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller` (p. 164), `activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller` (p. 200), `activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller` (p. 302), `activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller` (p. 325), `activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller` (p. 365), `activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller` (p. 453), `activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller` (p. 547), `activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller` (p. 739), `activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller` (p. 1077), `activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller` (p. 1101), `activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller` (p. 1152), `activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller` (p. 1188), `activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller` (p. 1240), `activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller` (p. 1264), `activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller` (p. 1288), `activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller` (p. 1328), `activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller` (p. 1397), `activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller` (p. 1488), `activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller` (p. 1577), `activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller` (p. 1671), `activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller` (p. 1817), `activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller` (p. 2062), `activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller` (p. 2097), `activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller` (p. 2122), `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2167), `activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller` (p. 2209), `activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller` (p. 2425), `activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller` (p. 2476), `activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller` (p. 2550), `activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller` (p. 2570), `activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller` (p. 2589), `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 2621), `activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller` (p. 2707), `activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller` (p. 2773), and `activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller` (p. 3058).

6.114.3.2 virtual void `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller::looseUnmarshal` (`OpenWireFormat * wireFormat`, `commands::DataStructure * dataStructure`, `decaf::io::DataInputStream * dataIn`) throw (`decaf::io::IOException`) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1348).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller` (p. 165), `activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller` (p. 200), `activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller` (p. 303), `activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller` (p. 325), `activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller` (p. 365), `activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller` (p. 454), `activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller` (p. 548), `activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller` (p. 740), `activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller` (p. 1078), `activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller` (p. 1101), `activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller` (p. 1153), `activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller` (p. 1189), `activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller` (p. 1241), `activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller` (p. 1264), `activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller` (p. 1289), `activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller` (p. 1328), `activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller` (p. 1397), `activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller` (p. 1489), `activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller` (p. 1578), `activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller` (p. 1672), `activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller` (p. 1818), `activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller` (p. 2062), `activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller` (p. 2098), `activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller` (p. 2122), `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2168), `activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller` (p. 2209), `activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller` (p. 2426), `activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller` (p. 2476), `activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller` (p. 2550), `activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller` (p. 2570), `activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller` (p. 2589), `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 2622), `activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller` (p. 2707), `activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller` (p. 2774), and `activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller` (p. 3059).

```
6.114.3.3 virtual int ac-
    tivemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller::tightMarshal1
    ( OpenWireFormat * wireFormat, commands::DataStructure
    * dataStructure, utils::BooleanStream * bs ) throw (
    decaf::io::IOException ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- bs* - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

- IOException* if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1354).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller` (p. 165), `activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller` (p. 200), `activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller` (p. 303), `activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller` (p. 325), `activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller` (p. 365), `activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller` (p. 454), `activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller` (p. 548), `activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller` (p. 740), `activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller` (p. 1078), `activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller` (p. 1101), `activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller` (p. 1153), `activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller` (p. 1189), `activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller` (p. 1241), `activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller` (p. 1264), `activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller` (p. 1289), `activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller` (p. 1328), `activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller` (p. 1398), `activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller` (p. 1489), `activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller` (p. 1578), `activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller` (p. 1672), `activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller` (p. 1818), `activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller` (p. 2063), `activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller` (p. 2098), `activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller` (p. 2123), `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2169), `activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller` (p. 2210), `activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller` (p. 2426), `activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller` (p. 2476), `activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller` (p. 2550), `activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller` (p. 2570), `activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller` (p. 2590), `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 2622), `activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller` (p. 2707), `activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller` (p. 2774), and `activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller` (p. 3059).

```

6.114.3.4 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1360).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller` (p. 165), `activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller` (p. 201), `activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller` (p. 303), `activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller` (p. 326), `activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller` (p. 366), `activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller` (p. 454), `activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller` (p. 548), `activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller` (p. 740), `activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller` (p. 1078), `activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller` (p. 1102), `activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller` (p. 1153), `activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller` (p. 1189), `activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller` (p. 1241), `activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller` (p. 1265), `activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller` (p. 1290), `activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller` (p. 1329), `activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller` (p. 1398), `activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller` (p. 1490), `activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller` (p. 1579), `activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller` (p. 1673), `activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller` (p. 1818), `activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller` (p. 2063), `activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller` (p. 2098), `activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller` (p. 2123), `activemq::wireformat::openwire::marshal::v2::MessageMarshaller` (p. 2169), `activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller` (p. 2210), `activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller` (p. 2426), `activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller` (p. 2477), `activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller` (p. 2551), `activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller` (p. 2571), `activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller` (p. 2590), `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 2623),

activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller (p. 2708), activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller (p. 2774), and activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller (p. 3059).

6.114.3.5 virtual void activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Implements activemq::wireformat::openwire::marshal::DataStreamMarshaller (p. 1366).

Reimplemented in activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller (p. 166), activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller (p. 201), activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller (p. 304), activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller (p. 326), activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller (p. 366), activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller (p. 455), activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller (p. 549), activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller (p. 741), activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller (p. 1079), activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller (p. 1102), activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller (p. 1154), activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller (p. 1190), activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller (p. 1242), activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller (p. 1265), activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller (p. 1290), activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller (p. 1329), activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller (p. 1399), activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller (p. 1490), activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller (p. 1579), activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller (p. 1673), activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller (p. 1819), activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller (p. 2063), activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller (p. 2099), activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller (p. 2123), activemq::wireformat::openwire::marshal::v2::MessageMarshaller (p. 2170), activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller (p. 2210), activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller (p. 2427), activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller (p. 2477),

`activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller` (p. 2551), `activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller` (p. 2571), `activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller` (p. 2590), `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 2623), `activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller` (p. 2708), `activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller` (p. 2775), and `activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller` (p. 3060).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h`

6.115 `activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller` Class Reference

Base class for all Marshallers that marshal DataStructures to and from the wire using the Open-Wire protocol.

```
#include <src/main/activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
```

Inherits `activemq::wireformat::openwire::marshal::DataStreamMarshaller`.

Inherited by `activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller`,

`activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller`,

`activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller`, ac-

`tivemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller`, ac-

`tivemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller`, ac-

`tivemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller`, ac-

`tivemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller`, ac-

`tivemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller`, ac-

`tivemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller`, ac-

`tivemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller`,

`activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller`, ac-

`tivemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller`,

`activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller`,

`activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller`, ac-

`tivemq::wireformat::openwire::marshal::v1::SessionIdMarshaller`, ac-

`tivemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller`, ac-

`tivemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller`, ac-

`tivemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller`, ac-

`tivemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller`,

`activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller`,

`activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller`, ac-

`tivemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller`, ac-

`tivemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller`, ac-

`tivemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller`, ac-

`tivemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller`, ac-

`tivemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller`, ac-

`tivemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller`, ac-

`tivemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller`,

`activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller`, ac-

`tivemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller`,

`activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller`,

`activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller`, ac-

tivemq::wireformat::openwire::marshal::v2::SessionIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller,	
activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller,	
activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller,	
activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller,	
activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller,	
activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v3::SessionIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller,	
activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller,	
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller,	
activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller,	
activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller,	
activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v4::SessionIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller,	
activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller,	
activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller,	
activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller,	ac-
tivemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller,	
activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller,	
activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller,	ac-

tivemq::wireformat::openwire::marshal::v5::SessionIdMarshaller, ac-
 tivemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller, ac-
 tivemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller, and ac-
 tivemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller.

Public Member Functions

- virtual `~BaseDataStreamMarshaller()`
- virtual `int tightMarshal (OpenWireFormat *format AMQCPP_UNUSED, commands::DataStructure *command AMQCPP_UNUSED, utils::BooleanStream *bs AMQCPP_UNUSED) throw (decaf::io::IOException)`
Tight Marshal to the given stream.
- virtual `void tightMarshal2 (OpenWireFormat *format AMQCPP_UNUSED, commands::DataStructure *command AMQCPP_UNUSED, decaf::io::DataOutputStream *ds AMQCPP_UNUSED, utils::BooleanStream *bs AMQCPP_UNUSED) throw (decaf::io::IOException)`
Tight Marshal to the given stream.
- virtual `void tightUnmarshal (OpenWireFormat *format AMQCPP_UNUSED, commands::DataStructure *command AMQCPP_UNUSED, decaf::io::DataInputStream *dis AMQCPP_UNUSED, utils::BooleanStream *bs AMQCPP_UNUSED) throw (decaf::io::IOException)`
Tight Un-Marshal to the given stream.
- virtual `void looseMarshal (OpenWireFormat *format AMQCPP_UNUSED, commands::DataStructure *command AMQCPP_UNUSED, decaf::io::DataOutputStream *ds AMQCPP_UNUSED) throw (decaf::io::IOException)`
Tight Marshal to the given stream.
- virtual `void looseUnmarshal (OpenWireFormat *format AMQCPP_UNUSED, commands::DataStructure *command AMQCPP_UNUSED, decaf::io::DataInputStream *dis AMQCPP_UNUSED) throw (decaf::io::IOException)`
Loose Un-Marshal to the given stream.

Static Public Member Functions

- static `std::string toString (const commands::MessageId *id)`
Converts the object to a String.
- static `std::string toString (const commands::ProducerId *id)`
Converts the object to a String.
- static `std::string toString (const commands::TransactionId *txnId)`
Converts the given transaction ID into a String.
- static `std::string toHexFromBytes (const std::vector< unsigned char > &data)`
given an array of bytes, convert that array to a Hexidecimal coded string that represents that data.

Protected Member Functions

- virtual **commands::DataStructure** * **tightUnmarshalCachedObject** (**OpenWireFormat** *wireFormat, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Tight Unmarshal the cached object.
- virtual int **tightMarshalCachedObject1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *data, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Tightly marshals the passed DataStructure based object to the passed BooleanStream returning the size of the data marshaled.
- virtual void **tightMarshalCachedObject2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *data, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Tightly marshals the passed DataStructure based object to the passed streams returning nothing.
- virtual void **looseMarshalCachedObject** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *data, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Loosely marshals the passed DataStructure based object to the passed stream returning nothing.
- virtual **commands::DataStructure** * **looseUnmarshalCachedObject** (**OpenWireFormat** *wireFormat, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Loose Unmarshal the cached object.
- virtual int **tightMarshalNestedObject1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *object, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Tightly marshals the passed DataStructure based object to the passed BooleanStream returning the size of the data marshaled.
- virtual void **tightMarshalNestedObject2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *object, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Tightly marshals the passed DataStructure based object to the passed streams returning nothing.
- virtual **commands::DataStructure** * **tightUnmarshalNestedObject** (**OpenWireFormat** *wireFormat, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Tight Unmarshal the nested object.
- virtual **commands::DataStructure** * **looseUnmarshalNestedObject** (**OpenWireFormat** *wireFormat, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Loose Unmarshal the nested object.
- virtual void **looseMarshalNestedObject** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *object, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)

Loose marshal the nested object.

- virtual std::string **tightUnmarshalString** (decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)

Performs Tight Unmarshaling of String Objects.

- virtual int **tightMarshalString1** (const std::string &value, utils::BooleanStream *bs) throw (decaf::io::IOException)

Tight Marshals the String to a Booleans Stream Object, returns the marshaled size.

- virtual void **tightMarshalString2** (const std::string &value, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)

Tight Marshals the passed string to the streams passed.

- virtual void **looseMarshalString** (const std::string value, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Loose Marshal the String to the DataOutputStream passed.

- virtual std::string **looseUnmarshalString** (decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Loose Un-Marshal the String to the DataOutputStream passed.

- virtual int **tightMarshalLong1** (OpenWireFormat *wireFormat, long long value, utils::BooleanStream *bs) throw (decaf::io::IOException)

Tightly marshal the long long to the BooleanStream passed.

- virtual void **tightMarshalLong2** (OpenWireFormat *wireFormat, long long value, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)

Tightly marshal the long long to the Streams passed.

- virtual long long **tightUnmarshalLong** (OpenWireFormat *wireFormat, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)

Tight marshal the long long type.

- virtual void **looseMarshalLong** (OpenWireFormat *wireFormat, long long value, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Tightly marshal the long long to the BooleanStream passed.

- virtual long long **looseUnmarshalLong** (OpenWireFormat *wireFormat, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Loose marshal the long long type.

- virtual std::vector< unsigned char > **tightUnmarshalByteArray** (decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)

Tight Unmarshal an array of char.

- virtual std::vector< unsigned char > **looseUnmarshalByteArray** (decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)
Loose Unmarshal an array of char.
- virtual std::vector< unsigned char > **tightUnmarshalConstByteArray** (decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs, int size) throw (decaf::io::IOException)
Tight Unmarshal a fixed size array from that data input stream and return an stl vector of char as the resultant.
- virtual std::vector< unsigned char > **looseUnmarshalConstByteArray** (decaf::io::DataInputStream *dataIn, int size) throw (decaf::io::IOException)
Tight Unmarshal a fixed size array from that data input stream and return an stl vector of char as the resultant.
- virtual commands::DataStructure * **tightUnmarshalBrokerError** (OpenWireFormat *wireFormat, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)
Tight Unmarshal the Error object.
- virtual int **tightMarshalBrokerError1** (OpenWireFormat *wireFormat, commands::DataStructure *data, utils::BooleanStream *bs) throw (decaf::io::IOException)
Tight Marshal the Error object.
- virtual void **tightMarshalBrokerError2** (OpenWireFormat *wireFormat, commands::DataStructure *data, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)
Tight Marshal the Error object.
- virtual commands::DataStructure * **looseUnmarshalBrokerError** (OpenWireFormat *wireFormat, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)
Loose Unmarshal the Error object.
- virtual void **looseMarshalBrokerError** (OpenWireFormat *wireFormat, commands::DataStructure *data, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)
Tight Marshal the Error object.
- template<typename T >
int **tightMarshalObjectArray1** (OpenWireFormat *wireFormat, std::vector< T > objects, utils::BooleanStream *bs) throw (decaf::io::IOException)
Tightly Marshal an array of DataStructure objects to the provided boolean stream, and return the size that the tight marshalling is going to take.
- template<typename T >
void **tightMarshalObjectArray2** (OpenWireFormat *wireFormat, std::vector< T > objects, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)
Tightly Marshal an array of DataStructure objects to the provided boolean stream and data output stream.

- `template<typename T >`
`void looseMarshalObjectArray (OpenWireFormat *wireFormat, std::vector< T > ob-`
`jects, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)`

Loosely Marshal an array of DataStructure objects to the provided boolean stream and data output stream.

- `virtual std::string readAsciiString (decaf::io::DataInputStream *dataIn) throw (de-`
`cafe::io::IOException)`

Given an DataInputStream read a know ASCII formatted string from the input and return that string.

6.115.1 Detailed Description

Base class for all Marshallers that marshal DataStructures to and from the wire using the Open-Wire protocol.

Since

2.0

6.115.2 Constructor & Destructor Documentation

- 6.115.2.1 `virtual`
`activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::~BaseDataStreamM`
`() [inline, virtual]`

6.115.3 Member Function Documentation

- 6.115.3.1 `virtual void ac-`
`tivemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshal`
`(OpenWireFormat *format AMQCPP_UNUSED, com-`
`mands::DataStructure *command AMQCPP_UNUSED,`
`decaf::io::DataOutputStream *ds AMQCPP_UNUSED) throw (`
`decaf::io::IOException) [inline, virtual]`

Tight Marshal to the given stream.

Parameters

format - The OpenwireFormat properties

command - the object to Marshal

ds - DataOutputStream to marshal to

Exceptions

IOException if an error occurs.

6.115.3.2 virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshalBrokerError
 (OpenWireFormat * *wireFormat*, commands::DataStructure * *data*,
 decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException
) [protected, virtual]

Tight Marshal the Error object.

Parameters

wireFormat - The OpenwireFormat properties

data - Error to Marshal

dataOut - stream to write marshalled data to

Exceptions

IOException if an error occurs.

6.115.3.3 virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshalCachedObject
 (OpenWireFormat * *wireFormat*, commands::DataStructure * *data*,
 decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException
) [protected, virtual]

Loosely marshals the passed DataStructure based object to the passed stream returning nothing.

Parameters

wireFormat - The OpenwireFormat properties

data - DataStructure Object Pointer to marshal

dataOut - stream to write marshaled data to

Exceptions

IOException if an error occurs.

6.115.3.4 virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshalLong
 (OpenWireFormat * *wireFormat*, long long *value*, de-
 caf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException)
 [protected, virtual]

Tightly marshal the long long to the BooleanStream passed.

Parameters

wireFormat - The OpenwireFormat properties

value - long long to marshal

dataOut - DataOutputStream to marshal to.

Exceptions

IOException if an error occurs.

6.115.3.5 `virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshalNestedObject (OpenWireFormat * wireFormat, commands::DataStructure * object, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [protected, virtual]

Loose marshal the nested object.

Parameters

wireFormat - The OpenwireFormat properties
object - DataStructure Object Pointer to marshal
dataOut - stream to write marshaled data to

Exceptions

IOException if an error occurs.

6.115.3.6 `template<typename T > void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshalObjectArray (OpenWireFormat * wireFormat, std::vector< T > objects, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [inline, protected]

Loosely Marshal an array of DataStructure objects to the provided boolean stream and data output stream.

Parameters

wireFormat - The OpenwireFormat properties
objects - array of DataStructure object pointers.
dataOut - stream to write marshalled data to

Returns

size of the marshalled data

Exceptions

IOException if an error occurs.

References `AMQ_CATCH_EXCEPTION_CONVERT`, `AMQ_CATCH_RETHROW`, and `AMQ_CATCHALL_THROW`.

6.115.3.7 `virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshalString (const std::string value, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [protected, virtual]

Loose Marshal the String to the DataOutputStream passed.

Parameters

value - string to marshal

dataOut - stream to write marshaled form to

Exceptions

IOException if an error occurs.

6.115.3.8 virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseUnmarshal (OpenWireFormat *format *AMQCPP_UNUSED*, commands::DataStructure *command *AMQCPP_UNUSED*, decaf::io::DataInputStream *dis *AMQCPP_UNUSED*) throw (decaf::io::IOException) [inline, virtual]

Loose Un-Marshal to the given stream.

Parameters

format - The OpenWireFormat properties
command - the object to Un-Marshal
dis - the DataInputStream to Un-Marshal from

Exceptions

IOException if an error occurs.

6.115.3.9 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseUnmarshalBroken (OpenWireFormat * *wireFormat*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [protected, virtual]

Loose Unmarshal the Error object.

Parameters

wireFormat - The OpenWireFormat properties
dataIn - stream to read marshalled form from

Returns

pointer to a new DataStructure Object

Exceptions

IOException if an error occurs.

6.115.3.10 virtual std::vector<unsigned char> activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseUnmarshalBytes (decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [protected, virtual]

Loose Unmarshal an array of char.

Parameters

dataIn - the DataInputStream to Un-Marshal from

Returns

the unmarshalled vector of chars.

Exceptions

IOException if an error occurs.

```
6.115.3.11 virtual commands::DataStructure* ac-
           tivemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseUnmarshalCachedObject(
           ( OpenWireFormat * wireFormat, decaf::io::DataInputStream *
           dataIn ) throw ( decaf::io::IOException ) [protected, virtual]
```

Loose Unmarshal the cached object.

Parameters

wireFormat - The OpenwireFormat properties

dataIn - stream to read marshaled form from

Returns

pointer to a new DataStructure Object

Exceptions

IOException if an error occurs.

```
6.115.3.12 virtual std::vector<unsigned char> ac-
           tivemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseUnmarshalConstrainedObject(
           ( decaf::io::DataInputStream * dataIn, int size ) throw (
           decaf::io::IOException ) [protected, virtual]
```

Tight Unmarshal a fixed size array from that data input stream and return an stl vector of char as the resultant.

Parameters

dataIn - the DataInputStream to Un-Marshal from

size - size of the const array to unmarshal

Returns

the unmarshalled vector of chars.

Exceptions

IOException if an error occurs.

6.115.3.13 `virtual long long activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseUnmarshalLong (OpenWireFormat * wireFormat, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [protected, virtual]

Loose marshal the long long type.

Parameters

wireFormat - The OpenwireFormat properties

dataIn - stream to read marshaled form from

Returns

the unmarshaled long long

Exceptions

IOException if an error occurs.

6.115.3.14 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseUnmarshalNested (OpenWireFormat * wireFormat, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [protected, virtual]

Loose Unmarshal the nested object.

Parameters

wireFormat - The OpenwireFormat properties

dataIn - stream to read marshaled form from

Returns

pointer to a new DataStructure Object

Exceptions

IOException if an error occurs.

6.115.3.15 `virtual std::string activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseUnmarshalString (decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [protected, virtual]

Loose Un-Marshal the String to the DataOutputStream passed.

Parameters

dataIn - stream to read marshaled form from

Returns

the unmarshaled string

Exceptions

IOException if an error occurs.

6.115.3.16 `virtual std::string activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::readAsciiString (decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [protected, virtual]

Given an DataInputStream read a know ASCII formatted string from the input and return that string.

Parameters

dataIn - DataInputStream to read from

Returns

string value read from stream

6.115.3.17 `virtual int activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshal1 (OpenWireFormat *format AMQCPP_UNUSED, commands::DataStructure *command AMQCPP_UNUSED, utils::BooleanStream *bs AMQCPP_UNUSED) throw (decaf::io::IOException)` [inline, virtual]

Tight Marshal to the given stream.

Parameters

format - The OpenWireFormat properties

command - the object to Marshal

bs - boolean stream to marshal to.

Exceptions

IOException if an error occurs.

6.115.3.18 `virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshal2 (OpenWireFormat *format AMQCPP_UNUSED, commands::DataStructure *command AMQCPP_UNUSED, decaf::io::DataOutputStream *ds AMQCPP_UNUSED, utils::BooleanStream *bs AMQCPP_UNUSED) throw (decaf::io::IOException)` [inline, virtual]

Tight Marshal to the given stream.

Parameters

format - The OpenWireFormat properties

command - the object to Marshal

ds - the DataOutputStream to Marshal to

bs - boolean stream to marshal to.

Exceptions

IOException if an error occurs.

6.115.3.19 virtual int activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalBroker(OpenWireFormat * *wireFormat*, commands::DataStructure * *data*, utils::BooleanStream * *bs*) throw (decaf::io::IOException)
[protected, virtual]

Tight Marshal the Error object.

Parameters

wireFormat - The OpenwireFormat properties

data - Error to Marshal

bs - boolean stream to marshal to.

Returns

size of the marshalled data

Exceptions

IOException if an error occurs.

6.115.3.20 virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalBroker(OpenWireFormat * *wireFormat*, commands::DataStructure * *data*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*)
throw (decaf::io::IOException) [protected, virtual]

Tight Marshal the Error object.

Parameters

wireFormat - The OpenwireFormat properties

data - Error to Marshal

dataOut - stream to write marshalled data to

bs - boolean stream to marshal to.

Exceptions

IOException if an error occurs.

6.115.3.21 virtual int activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalCached(OpenWireFormat * *wireFormat*, commands::DataStructure * *data*, utils::BooleanStream * *bs*) throw (decaf::io::IOException)
[protected, virtual]

Tightly marshals the passed DataStructure based object to the passed BooleanStream returning the size of the data marshaled.

Parameters

wireFormat - The OpenwireFormat properties

data - DataStructure Object Pointer to marshal

bs - boolean stream to marshal to.

Returns

size of data written.

Exceptions

IOException if an error occurs.

6.115.3.22 `virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalCached(OpenWireFormat * wireFormat, commands::DataStructure * data, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [protected, virtual]

Tightly marshals the passed DataStructure based object to the passed streams returning nothing.

Parameters

wireFormat - The OpenWireFormat properties

data - DataStructure Object Pointer to marshal

bs - boolean stream to marshal to.

dataOut - stream to write marshaled data to

Exceptions

IOException if an error occurs.

6.115.3.23 `virtual int activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalLong1(OpenWireFormat * wireFormat, long long value, utils::BooleanStream * bs) throw (decaf::io::IOException)` [protected, virtual]

Tightly marshal the long long to the BooleanStream passed.

Parameters

wireFormat - The OpenWireFormat properties

value - long long to marshal

bs - boolean stream to marshal to.

Returns

size of data written.

Exceptions

IOException if an error occurs.

```

6.115.3.24  virtual void ac-
            tivemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalLong2
            ( OpenWireFormat * wireFormat, long long value,
              decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs )
            throw ( decaf::io::IOException ) [protected, virtual]

```

Tightly marshal the long long to the Streams passed.

Parameters

wireFormat - The OpenwireFormat properties

value - long long to marshal

dataOut - stream to write marshaled form to

bs - boolean stream to marshal to.

Exceptions

IOException if an error occurs.

```

6.115.3.25  virtual int ac-
            tivemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalNested
            ( OpenWireFormat * wireFormat, commands::DataStructure *
              object, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
            [protected, virtual]

```

Tightly marshals the passed DataStructure based object to the passed BooleanStream returning the size of the data marshaled.

Parameters

wireFormat - The OpenwireFormat properties

object - DataStructure Object Pointer to marshal

bs - boolean stream to marshal to.

Returns

size of data written.

Exceptions

IOException if an error occurs.

```

6.115.3.26  virtual void ac-
            tivemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalNested
            ( OpenWireFormat * wireFormat, commands::DataStructure *
              object, decaf::io::DataOutputStream * dataOut, utils::BooleanStream
              * bs ) throw ( decaf::io::IOException ) [protected, virtual]

```

Tightly marshals the passed DataStructure based object to the passed streams returning nothing.

Parameters

wireFormat - The OpenwireFormat properties

object - DataStructure Object Pointer to marshal

bs - boolean stream to marshal to.

dataOut - stream to write marshaled data to

Exceptions

IOException if an error occurs.

```
6.115.3.27  template<typename T > int ac-
             tivemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalObject
             ( OpenWireFormat * wireFormat, std::vector< T > objects,
             utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [inline,
             protected]
```

Tightly Marshal an array of DataStructure objects to the provided boolean stream, and return the size that the tight marshalling is going to take.

Parameters

wireFormat - The OpenwireFormat properties

objects - array of DataStructure object pointers.

bs - boolean stream to marshal to.

Returns

size of the marshalled data

Exceptions

IOException if an error occurs.

References AMQ_CATCH_EXCEPTION_CONVERT, AMQ_CATCH_RETHROW, and AMQ_CATCHALL_THROW.

```
6.115.3.28  template<typename T > void ac-
             tivemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalObject
             ( OpenWireFormat * wireFormat, std::vector< T > objects,
             decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs )
             throw ( decaf::io::IOException ) [inline, protected]
```

Tightly Marshal an array of DataStructure objects to the provided boolean stream and data output stream.

Parameters

wireFormat - The OpenwireFormat properties

objects - array of DataStructure object pointers.

dataOut - stream to write marshalled data to

bs - boolean stream to marshal to.

Returns

size of the marshalled data

Exceptions

IOException if an error occurs.

References AMQ_CATCH_EXCEPTION_CONVERT, AMQ_CATCH_RETHROW, and AMQ_CATCHALL_THROW.

6.115.3.29 `virtual int activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalString1 (const std::string & value, utils::BooleanStream * bs) throw (decaf::io::IOException)` [protected, virtual]

Tight Marshals the String to a Booleans Stream Object, returns the marshaled size.

Parameters

value - string to marshal
bs - BooleanStream to use.

Returns

size of marshaled string.

Exceptions

IOException if an error occurs.

6.115.3.30 `virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalString2 (const std::string & value, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [protected, virtual]

Tight Marshals the passed string to the streams passed.

Parameters

value - string to marshal
dataOut - the DataOutputStream to Marshal to
bs - boolean stream to marshal to.

Exceptions

IOException if an error occurs.

6.115.3.31 `virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightUnmarshal (OpenWireFormat *format AMQCPP_UNUSED, commands::DataStructure *command AMQCPP_UNUSED, decaf::io::DataInputStream *dis AMQCPP_UNUSED, utils::BooleanStream *bs AMQCPP_UNUSED) throw (decaf::io::IOException)` [inline, virtual]

Tight Un-Marshal to the given stream.

Parameters

format - The OpenwireFormat properties
command - the object to Un-Marshall
dis - the DataInputStream to Un-Marshall from
bs - boolean stream to Un-Marshall from.

Exceptions

IOException if an error occurs.

```
6.115.3.32  virtual commands::DataStructure* ac-
             tivemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightUnmarshalBroken
             ( OpenWireFormat * wireFormat, decaf::io::DataInputStream *
               dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
             [protected, virtual]
```

Tight Unmarshal the Error object.

Parameters

wireFormat - The OpenwireFormat properties
dataIn - stream to read marshalled form from
bs - boolean stream to marshal to.

Returns

pointer to a new DataStructure Object

Exceptions

IOException if an error occurs.

```
6.115.3.33  virtual std::vector<unsigned char> ac-
             tivemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightUnmarshalByte
             ( decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs )
             throw ( decaf::io::IOException ) [protected, virtual]
```

Tight Unmarshal an array of char.

Parameters

dataIn - the DataInputStream to Un-Marshall from
bs - boolean stream to unmarshal from.

Returns

the unmarshaled vector of chars.

Exceptions

IOException if an error occurs.

6.115.3.34 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightUnmarshalCached (OpenWireFormat * wireFormat, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException)` [protected, virtual]

Tight Unmarshal the cached object.

Parameters

wireFormat - The OpenwireFormat properties

dataIn - stream to read marshaled form from

bs - boolean stream to marshal to.

Returns

pointer to a new DataStructure Object

Exceptions

IOException if an error occurs.

6.115.3.35 `virtual std::vector<unsigned char> activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightUnmarshalConst (decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs, int size) throw (decaf::io::IOException)` [protected, virtual]

Tight Unmarshal a fixed size array from that data input stream and return an stl vector of char as the resultant.

Parameters

dataIn - the DataInputStream to Un-Marshall from

bs - boolean stream to unmarshal from.

size - size of the const array to unmarshal

Returns

the unmarshaled vector of chars.

Exceptions

IOException if an error occurs.

6.115.3.36 `virtual long long activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightUnmarshalLong (OpenWireFormat * wireFormat, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException)` [protected, virtual]

Tight marshal the long long type.

Parameters

wireFormat - The OpenwireFormat properties

dataIn - stream to read marshaled form from

bs - boolean stream to marshal to.

Returns

the unmarshaled long long

Exceptions

IOException if an error occurs.

```
6.115.3.37  virtual commands::DataStructure* ac-
            tivemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightUnmarshalNest
            ( OpenWireFormat * wireFormat, decaf::io::DataInputStream *
              dataIn, utils::BooleanStream * bs ) throw ( decaf::io::IOException )
            [protected, virtual]
```

Tight Unmarshal the nested object.

Parameters

wireFormat - The OpenwireFormat properties

dataIn - stream to read marshaled form from

bs - boolean stream to marshal to.

Returns

pointer to a new DataStructure Object

Exceptions

IOException if an error occurs.

```
6.115.3.38  virtual std::string ac-
            tivemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightUnmarshalStrin
            ( decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs )
            throw ( decaf::io::IOException ) [protected, virtual]
```

Performs Tight Unmarshaling of String Objects.

Parameters

dataIn - the DataInputStream to Un-Marshal from

bs - boolean stream to unmarshal from.

Returns

the unmarshaled string.

Exceptions

IOException if an error occurs.

6.115.3.39 `static std::string activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::toHexFromBytes (const std::vector< unsigned char > & data) [static]`

given an array of bytes, convert that array to a Hexidecimal coded string that represents that data.

Parameters

data - unsigned char data array pointer

Returns

a string coded in hex that represents the data

6.115.3.40 `static std::string activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::toString (const commands::TransactionId * txnId) [static]`

Converts the given transaction ID into a String.

Parameters

txnId - TransactionId poitner

Returns

string representation of the id

6.115.3.41 `static std::string activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::toString (const commands::ProducerId * id) [static]`

Converts the object to a String.

Parameters

id - ProducerId pointer

Returns

string representing the id

6.115.3.42 `static std::string activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::toString (const commands::MessageId * id) [static]`

Converts the object to a String.

Parameters

id - MessageId pointer

Returns

string representing the id

Referenced by `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getCMSMessageID()`.

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h`

6.116 `activemq::commands::BaseDataStructure` Class Reference

```
#include <src/main/activemq/commands/BaseDataStructure.h>
```

Inheritance diagram for `activemq::commands::BaseDataStructure`:

Public Member Functions

- virtual `~BaseDataStructure()`
- virtual bool `isMarshalAware()` const
Determine if this object is aware of marshaling and should have its before and after marshaling methods called.
- virtual void `beforeMarshal(wireformat::WireFormat *wireFormat AMQCPP_UNUSED)` throw (`decaf::io::IOException`)
Perform any processing needed before an marshal.
- virtual void `afterMarshal(wireformat::WireFormat *wireFormat AMQCPP_UNUSED)` throw (`decaf::io::IOException`)
Perform any processing needed after an unmarshal.
- virtual void `beforeUnmarshal(wireformat::WireFormat *wireFormat AMQCPP_UNUSED)` throw (`decaf::io::IOException`)
Perform any processing needed before an unmarshal.
- virtual void `afterUnmarshal(wireformat::WireFormat *wireFormat AMQCPP_UNUSED)` throw (`decaf::io::IOException`)
Perform any processing needed after an unmarshal.
- virtual void `setMarshaledForm(wireformat::WireFormat *wireFormat AMQCPP_UNUSED, const std::vector< char > &data AMQCPP_UNUSED)`
Called to set the data to this object that will contain the objects marshaled form.
- virtual `std::vector< unsigned char > getMarshaledForm(wireformat::WireFormat *wireFormat AMQCPP_UNUSED)`
Called to get the data to this object that will contain the objects marshaled form.

- virtual void **copyDataStructure** (const **DataStructure** *src AMQCPP_UNUSED)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1372) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value AMQCPP_UNUSED) const
*Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent.*

6.116.1 Constructor & Destructor Documentation

- 6.116.1.1** virtual **activemq::commands::BaseDataStructure::~~BaseDataStructure** (
) [inline, virtual]

6.116.2 Member Function Documentation

- 6.116.2.1** virtual void **activemq::commands::BaseDataStructure::afterMarshal** (
 wireformat::WireFormat *wireFormat AMQCPP_UNUSED) throw (
 decaf::io::IOException) [inline, virtual]

Perform any processing needed after an unmarshal.

Parameters

wireformat - the OpenWireFormat object in use.

- 6.116.2.2** virtual void **activemq::commands::BaseDataStructure::afterUnmarshal** (
 wireformat::WireFormat *wireFormat AMQCPP_UNUSED) throw (
 decaf::io::IOException) [inline, virtual]

Perform any processing needed after an unmarshal.

Parameters

wireformat - the OpenWireFormat object in use.

Reimplemented in **activemq::commands::Message** (p. 2022), and **activemq::commands::WireFormatInfo** (p. 3155).

- 6.116.2.3** virtual void **activemq::commands::BaseDataStructure::beforeMarshal** (
 wireformat::WireFormat *wireFormat AMQCPP_UNUSED) throw (
 decaf::io::IOException) [inline, virtual]

Perform any processing needed before an marshal.

Parameters

wireformat - the OpenWireFormat object in use.

Reimplemented in **activemq::commands::Message** (p. 2022), and **activemq::commands::WireFormatInfo** (p. 3155).

6.116.2.4 `virtual void activemq::commands::BaseDataStructure::beforeUnmarshal
(wireformat::WireFormat *wireFormat AMQCPP_UNUSED) throw
(decaf::io::IOException) [inline, virtual]`

Perform any processing needed before an unmarshal.

Parameters

wireformat - the OpenWireFormat object in use.

6.116.2.5 `virtual void activemq::commands::BaseDataStructure::copyDataStructure
(const DataStructure *src AMQCPP_UNUSED) [inline, virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Reimplemented in `activemq::commands::BooleanExpression` (p. 680).

6.116.2.6 `virtual bool activemq::commands::BaseDataStructure::equals (const
DataStructure *value AMQCPP_UNUSED) const [inline, virtual]`

Compares the `DataStructure` (p. 1372) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Referenced by `activemq::commands::BooleanExpression::equals()`, and `activemq::commands::BaseCommand::equals()`.

6.116.2.7 `virtual std::vector<unsigned char> ac-
tivemq::commands::BaseDataStructure::getMarshaledForm
(wireformat::WireFormat *wireFormat AMQCPP_UNUSED)
[inline, virtual]`

Called to get the data to this object that will contain the objects marshaled form.

Parameters

wireFormat - the wireformat object to control unmarshaling

Returns

buffer that holds the objects data.

6.116.2.8 virtual bool activemq::commands::BaseDataStructure::isMarshalAware () const [inline, virtual]

Determine if this object is aware of marshaling and should have its before and after marshaling methods called.

Defaults to false.

Returns

true if aware of marshaling

Implements **activemq::wireformat::MarshalAware** (p. 1995).

Reimplemented in **activemq::commands::ActiveMQMapMessage** (p. 280), **activemq::commands::Message** (p. 2029), and **activemq::commands::WireFormatInfo** (p. 3158).

6.116.2.9 virtual void activemq::commands::BaseDataStructure::setMarshaledForm (wireformat::WireFormat *wireFormat *AMQCPP_UNUSED*, const std::vector< char > &data *AMQCPP_UNUSED*) [inline, virtual]

Called to set the data to this object that will contain the objects marshaled form.

Parameters

wireFormat - the wireformat object to control unmarshaling

data - vector of object binary data

6.116.2.10 virtual std::string activemq::commands::BaseDataStructure::toString () const [inline, virtual]

Returns a string containing the information for this **DataStructure** (p. 1372) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Implements **activemq::commands::DataStructure** (p. 1376).

Reimplemented in **activemq::commands::ActiveMQBlobMessage** (p. 146), **activemq::commands::ActiveMQBytesMessage** (p. 177), **activemq::commands::ActiveMQDestination** (p. 248), **activemq::commands::ActiveMQMapMessage** (p. 284), **activemq::commands::ActiveMQMessage** (p. 307), **activemq::commands::ActiveMQObjectMessage** (p. 346), **activemq::commands::ActiveMQStreamMessage** (p. 432), **activemq::commands::ActiveMQTextMessage** (p. 529), **activemq::commands::BaseCommand** (p. 602), **activemq::commands::BooleanExpression** (p. 680), **activemq::commands::BrokerInfo** (p. 719), **activemq::commands::Command** (p. 995), **activemq::commands::ConnectionControl** (p. 1058), **activemq::commands::ConnectionError** (p. 1082), **activemq::commands::ConnectionInfo** (p. 1133), **activemq::commands::ConsumerControl** (p. 1169), **activemq::commands::ConsumerInfo** (p. 1220), **activemq::commands::ControlCommand**

(p. 1245), `activemq::commands::DataArrayResponse` (p. 1270), `activemq::commands::DataResponse` (p. 1309), `activemq::commands::DestinationInfo` (p. 1394), `activemq::commands::DiscoveryEvent` (p. 1419), `activemq::commands::ExceptionResponse` (p. 1486), `activemq::commands::FlushCommand` (p. 1575), `activemq::commands::IntegerResponse` (p. 1669), `activemq::commands::JournalQueueAck` (p. 1721), `activemq::commands::JournalTopicAck` (p. 1745), `activemq::commands::JournalTrace` (p. 1769), `activemq::commands::JournalTransaction` (p. 1792), `activemq::commands::KeepAliveInfo` (p. 1815), `activemq::commands::LastPartialCommand` (p. 1842), `activemq::commands::Message` (p. 2033), `activemq::commands::MessageAck` (p. 2059), `activemq::commands::MessageDispatch` (p. 2087), `activemq::commands::MessageDispatchNotification` (p. 2119), `activemq::commands::MessagePull` (p. 2206), `activemq::commands::NetworkBridgeFilter` (p. 2249), `activemq::commands::PartialCommand` (p. 2321), `activemq::commands::ProducerAck` (p. 2423), `activemq::commands::ProducerInfo` (p. 2473), `activemq::commands::RemoveInfo` (p. 2539), `activemq::commands::RemoveSubscriptionInfo` (p. 2563), `activemq::commands::ReplayCommand` (p. 2586), `activemq::commands::Response` (p. 2614), `activemq::commands::SessionInfo` (p. 2704), `activemq::commands::ShutdownInfo` (p. 2759), `activemq::commands::SubscriptionInfo` (p. 2910), `activemq::commands::TransactionInfo` (p. 3040), and `activemq::commands::WireFormatInfo` (p. 3162).

Referenced by `activemq::commands::BooleanExpression::toString()`, and `activemq::commands::BaseCommand::toString()`.

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/BaseDataStructure.h`

6.117 `binary_function` Class Reference

Inheritance diagram for `binary_function`:

The documentation for this class was generated from the following file:

- `src/main/decaf/util/Comparator.h`

6.118 `decaf::net::BindException` Class Reference

```
#include <src/main/decaf/net/BindException.h>
```

Inheritance diagram for `decaf::net::BindException`:

Public Member Functions

- `BindException () throw ()`

Default Constructor.

- **BindException** (const Exception &ex) throw ()
Conversion Constructor from some other Exception.
- **BindException** (const **BindException** &ex) throw ()
Copy Constructor.
- **BindException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **BindException** (const std::exception *cause) throw ()
Constructor.
- **BindException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **BindException** * **clone** () const
Clones this exception.
- virtual ~**BindException** () throw ()

6.118.1 Constructor & Destructor Documentation

6.118.1.1 decaf::net::BindException::BindException () throw () [inline]

Default Constructor.

6.118.1.2 decaf::net::BindException::BindException (const Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

Parameters

ex An exception that should become this type of Exception

6.118.1.3 decaf::net::BindException::BindException (const BindException & ex) throw () [inline]

Copy Constructor.

Parameters

ex An exception that should become this type of Exception

6.118.1.4 `decaf::net::BindException::BindException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.118.1.5 `decaf::net::BindException::BindException (const std::exception * cause) throw () [inline]`

Constructor.

Parameters

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.118.1.6 `decaf::net::BindException::BindException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.118.1.7 `virtual decaf::net::BindException::~~BindException () throw () [inline, virtual]`

6.118.2 Member Function Documentation

6.118.2.1 `virtual BindException* decaf::net::BindException::clone () const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::net::SocketException** (p. 2794).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/BindException.h`

6.119 decaf::io::BlockingByteArrayInputStream Class Reference

This is a blocking version of a byte buffer stream.

```
#include <src/main/decaf/io/BlockingByteArrayInputStream.h>
```

Inheritance diagram for decaf::io::BlockingByteArrayInputStream:

Public Member Functions

- **BlockingByteArrayInputStream** ()
Default Constructor - uses a default internal buffer.
- **BlockingByteArrayInputStream** (const unsigned char *buffer, std::size_t bufferSize)
Constructor that initializes the internal buffer.
- virtual **~BlockingByteArrayInputStream** ()
Destructor.
- virtual void **setByteArray** (const unsigned char *buffer, std::size_t bufferSize)
Sets the data that this reader uses.
- virtual std::size_t **available** () const throw (IOException)
Indicates the number of bytes available to be read without blocking.
- virtual int **read** () throw (IOException)
Reads a single byte from the buffer.
- virtual int **read** (unsigned char *buffer, std::size_t offset, std::size_t bufferSize) throw (IOException, lang::exceptions::NullPointerException)
Reads an array of bytes from the buffer.
- virtual void **close** () throw (io::IOException)
Closes the target input stream.
- virtual std::size_t **skip** (std::size_t num) throw (io::IOException, lang::exceptions::UnsupportedOperationException)
Skips over and discards n bytes of data from this input stream.

- virtual void **mark** (int readLimit DECAF_UNUSED)
Marks the current position in the stream. A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.
- virtual void **reset** () throw (IOException)
Repositions this stream to the position at the time the mark method was last called on this input stream.
- virtual bool **markSupported** () const
Determines if this input stream supports the mark and reset methods.
- virtual void **lock** () throw (decaf::lang::exceptions::RuntimeException)
Locks the object.
- virtual bool **tryLock** () throw (decaf::lang::exceptions::RuntimeException)
Attempts to Lock the object, if the lock is already held by another thread than this method returns false.
- virtual void **unlock** () throw (decaf::lang::exceptions::RuntimeException)
Unlocks the object.
- virtual void **wait** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (long long millisecs) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (long long millisecs, int nanos) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **notify** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)
Signals a waiter on this object that it can now wake up and continue.
- virtual void **notifyAll** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)
Signals the waiters on this object that it can now wake up and continue.

6.119.1 Detailed Description

This is a blocking version of a byte buffer stream. Read operations block until the requested data becomes available in the internal buffer via a call to setByteArray.

6.119.2 Constructor & Destructor Documentation

6.119.2.1 decaf::io::BlockingByteArrayInputStream::BlockingByteArrayInputStream ()

Default Constructor - uses a default internal buffer.

6.119.2.2 decaf::io::BlockingByteArrayInputStream::BlockingByteArrayInputStream (const unsigned char * *buffer*, std::size_t *bufferSize*)

Constructor that initializes the internal buffer.

See also

`setByteArray` (p. 671).

6.119.2.3 virtual decaf::io::BlockingByteArrayInputStream::~~BlockingByteArrayInputStream () [virtual]

Destructor.

6.119.3 Member Function Documentation

6.119.3.1 virtual std::size_t decaf::io::BlockingByteArrayInputStream::available () const throw (IOException) [inline, virtual]

Indicates the number of bytes available to be read without blocking.

Returns

the data available in the internal buffer.

Exceptions

IOException (p. 1707) if an error occurs.

Implements `decaf::io::InputStream` (p. 1631).

6.119.3.2 virtual void decaf::io::BlockingByteArrayInputStream::close () throw (io::IOException) [virtual]

Closes the target input stream.

Exceptions

IOException (p. 1707) if an error occurs.

6.119.3.3 `virtual void decaf::io::BlockingByteArrayInputStream::lock () throw (decaf::lang::exceptions::RuntimeException) [inline, virtual]`

Locks the object.

Exceptions

RuntimeException if an error occurs while locking the object.

Implements `decaf::util::concurrent::Synchronizable` (p. 2932).

6.119.3.4 `virtual void decaf::io::BlockingByteArrayInputStream::mark (int readLimit DECAF_UNUSED) [inline, virtual]`

Marks the current position in the stream. A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.

If a stream instance reports that marks are supported then the stream will ensure that the same bytes can be read again after the reset method is called so long the readLimit is not reached.

Parameters

readLimit - max bytes read before marked position is invalid.

Implements `decaf::io::InputStream` (p. 1631).

6.119.3.5 `virtual bool decaf::io::BlockingByteArrayInputStream::markSupported () const [inline, virtual]`

Determines if this input stream supports the mark and reset methods.

Whether or not mark and reset are supported is an invariant property of a particular input stream instance.

Returns

true if this stream instance supports marks

Implements `decaf::io::InputStream` (p. 1631).

6.119.3.6 `virtual void decaf::io::BlockingByteArrayInputStream::notify () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException) [inline, virtual]`

Signals a waiter on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

RuntimeException if an error occurs while notifying one of the waiting threads.

Implements `decaf::util::concurrent::Synchronizable` (p. 2933).

6.119.3.7 `virtual void decaf::io::BlockingByteArrayInputStream::notifyAll () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException) [inline, virtual]`

Signals the waiters on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

RuntimeException if an error occurs while notifying the waiting threads.

Implements `decaf::util::concurrent::Synchronizable` (p. 2934).

6.119.3.8 `virtual int decaf::io::BlockingByteArrayInputStream::read (unsigned char * buffer, std::size_t offset, std::size_t bufferSize) throw (IOException, lang::exceptions::NullPointerException) [virtual]`

Reads an array of bytes from the buffer.

If the desired amount of data is not currently available, this operation will block until the appropriate amount of data is available in the buffer via a call to `setByteArray`.

Parameters

buffer (out) the target buffer

offset the position in the buffer to start from.

bufferSize the size of the output buffer.

Returns

the number of bytes read. or -1 if EOF

Exceptions

IOException (p. 1707) if an error occurs.

Implements `decaf::io::InputStream` (p. 1632).

6.119.3.9 `virtual int decaf::io::BlockingByteArrayInputStream::read () throw (IOException) [virtual]`

Reads a single byte from the buffer.

This operation will block until data has been added to the buffer via a call to `setByteArray`.

Returns

the next byte.

Exceptions

IOException (p. 1707) if an error occurs.

Implements `decaf::io::InputStream` (p. 1632).

6.119.3.10 `virtual void decaf::io::BlockingByteArrayInputStream::reset () throw (IOException) [inline, virtual]`

Repositions this stream to the position at the time the mark method was last called on this input stream.

If the method `markSupported` returns true, then: * If the method `mark` has not been called since the stream was created, or the number of bytes read from the stream since `mark` was last called is larger than the argument to `mark` at that last call, then an **IOException** (p. 1707) might be thrown. * If such an **IOException** (p. 1707) is not thrown, then the stream is reset to a state such that all the bytes read since the most recent call to `mark` (or since the start of the file, if `mark` has not been called) will be resupplied to subsequent callers of the `read` method, followed by any bytes that otherwise would have been the next input data as of the time of the call to `reset`. If the method `markSupported` returns false, then: * The call to `reset` may throw an **IOException** (p. 1707). * If an **IOException** (p. 1707) is not thrown, then the stream is reset to a fixed state that depends on the particular type of the input stream and how it was created. The bytes that will be supplied to subsequent callers of the `read` method depend on the particular type of the input stream.

Exceptions

IOException (p. 1707)

Implements `decaf::io::InputStream` (p. 1633).

6.119.3.11 `virtual void decaf::io::BlockingByteArrayInputStream::setByteArray (const unsigned char * buffer, std::size_t bufferSize) [virtual]`

Sets the data that this reader uses.

Replaces any existing data and resets the read index to the beginning of the buffer. When this method is called, it notifies any other threads that data is now available to be read.

Parameters

buffer The new data to be copied to the internal buffer.

bufferSize The size of the new buffer.

6.119.3.12 `virtual std::size_t decaf::io::BlockingByteArrayInputStream::skip (std::size_t num) throw (io::IOException, lang::exceptions::UnsupportedOperationException) [virtual]`

Skips over and discards *n* bytes of data from this input stream.

The `skip` method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before *n* bytes have been skipped is only one possibility. The actual number of bytes skipped is returned. If *n* is negative, no bytes are skipped.

The `skip` method of **InputStream** (p. 1630) creates a byte array and then repeatedly reads into it until *n* bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.

Parameters

num - the number of bytes to skip

Returns

total bytes skipped

Exceptions

IOException (p. 1707) if an error occurs

Implements **decaf::io::InputStream** (p. 1633).

6.119.3.13 **virtual bool decaf::io::BlockingByteArrayInputStream::tryLock ()
 throw (decaf::lang::exceptions::RuntimeException) [inline, virtual]**

Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

Returns

true if the lock was acquired, false if it is already held by another thread.

Exceptions

RuntimeException if an error occurs while locking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2935).

6.119.3.14 **virtual void decaf::io::BlockingByteArrayInputStream::unlock ()
 throw (decaf::lang::exceptions::RuntimeException) [inline, virtual]**

Unlocks the object.

Exceptions

RuntimeException if an error occurs while unlocking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2936).

6.119.3.15 **virtual void decaf::io::BlockingByteArrayInputStream::wait
 () throw (decaf::lang::exceptions::RuntimeException,
 decaf::lang::exceptions::IllegalMonitorStateException,
 decaf::lang::exceptions::InterruptedException) [inline, virtual]**

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling.

Exceptions

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2937).

```

6.119.3.16  virtual void decaf::io::BlockingByteArrayInputStream::wait
            ( long long millisecs, int nanos ) throw
            ( decaf::lang::exceptions::RuntimeException,
              decaf::lang::exceptions::IllegalArgumentException,
              decaf::lang::exceptions::IllegalMonitorStateException,
              decaf::lang::exceptions::InterruptedException ) [inline, virtual]

```

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

Parameters

millisecs the time in milliseconds to wait, or WAIT_INFINITE

nanos additional time in nanoseconds with a range of 0-999999

Exceptions

IllegalArgumentException if an error occurs or the nanos argument is not in the range of [0-999999]

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2940).

```

6.119.3.17  virtual void decaf::io::BlockingByteArrayInputStream::wait ( long
            long millisecs ) throw ( decaf::lang::exceptions::RuntimeException,
            decaf::lang::exceptions::IllegalMonitorStateException,
            decaf::lang::exceptions::InterruptedException ) [inline, virtual]

```

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval.

Parameters

millisecs the time in milliseconds to wait, or WAIT_INFINITE

Exceptions

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2938).

The documentation for this class was generated from the following file:

- `src/main/decaf/io/BlockingByteArrayInputStream.h`

6.120 decaf::lang::Boolean Class Reference

```
#include <src/main/decaf/lang/Boolean.h>
```

Inheritance diagram for `decaf::lang::Boolean`:

Public Member Functions

- **Boolean** (bool value)
- **Boolean** (const std::string &value)
- virtual **~Boolean** ()
- bool **booleanValue** () const
- std::string **toString** () const
- virtual int **compareTo** (const **Boolean** &b) const
*Compares this **Boolean** (p. 674) instance with another.*
- virtual bool **operator==** (const **Boolean** &value) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const **Boolean** &value) const
Compares this object to another and returns true if this object is considered to be less than the one passed.
- bool **equals** (const **Boolean** &b) const
- virtual int **compareTo** (const bool &b) const
*Compares this **Boolean** (p. 674) instance with another.*
- virtual bool **operator==** (const bool &value) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const bool &value) const
Compares this object to another and returns true if this object is considered to be less than the one passed.
- bool **equals** (const bool &b) const

Static Public Member Functions

- static **Boolean** **valueOf** (bool value)
- static **Boolean** **valueOf** (const std::string &value)
- static bool **parseBoolean** (const std::string &value)
Parses the String passed and extracts an bool.
- static std::string **toString** (bool value)
Converts the bool to a String representation.

Static Public Attributes

- static const **Boolean** `_FALSE`

The Class object representing the primitive false boolean.

- static const **Boolean** `_TRUE`

The Class object representing the primitive type boolean.

6.120.1 Constructor & Destructor Documentation

6.120.1.1 `decaf::lang::Boolean::Boolean (bool value)`

Parameters

value - primitive boolean to wrap.

6.120.1.2 `decaf::lang::Boolean::Boolean (const std::string & value)`

Parameters

value - String value to convert to a boolean.

6.120.1.3 `virtual decaf::lang::Boolean::~~Boolean ()` [inline, virtual]

6.120.2 Member Function Documentation

6.120.2.1 `bool decaf::lang::Boolean::booleanValue ()` const [inline]

Returns

the primitive boolean value of this object

6.120.2.2 `virtual int decaf::lang::Boolean::compareTo (const Boolean & b)` const [virtual]

Compares this **Boolean** (p. 674) instance with another.

Parameters

b - the **Boolean** (p. 674) instance to be compared

Returns

zero if this object represents the same boolean value as the argument; a positive value if this object represents true and the argument represents false; and a negative value if this object represents false and the argument represents true

6.120.2.3 `virtual int decaf::lang::Boolean::compareTo (const bool & b) const`
[virtual]

Compares this **Boolean** (p. 674) instance with another.

Parameters

b - the **Boolean** (p. 674) instance to be compared

Returns

zero if this object represents the same boolean value as the argument; a positive value if this object represents true and the argument represents false; and a negative value if this object represents false and the argument represents true

Implements **decaf::lang::Comparable**< **bool** > (p. 1010).

6.120.2.4 `bool decaf::lang::Boolean::equals (const bool & b) const` [inline,
virtual]

Returns

true if the two **Boolean** (p. 674) Objects have the same value.

Implements **decaf::lang::Comparable**< **bool** > (p. 1010).

6.120.2.5 `bool decaf::lang::Boolean::equals (const Boolean & b) const` [inline]

Returns

true if the two **Boolean** (p. 674) Objects have the same value.

6.120.2.6 `virtual bool decaf::lang::Boolean::operator< (const bool & value)`
`const` [virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

Parameters

value - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable**< **bool** > (p. 1011).

6.120.2.7 `virtual bool decaf::lang::Boolean::operator< (const Boolean & value)`
`const` [virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

Parameters

value - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

6.120.2.8 `virtual bool decaf::lang::Boolean::operator==(const bool & value) const [virtual]`

Compares equality between this object and the one passed.

Parameters

value - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

Implements `decaf::lang::Comparable< bool >` (p. 1011).

6.120.2.9 `virtual bool decaf::lang::Boolean::operator==(const Boolean & value) const [virtual]`

Compares equality between this object and the one passed.

Parameters

value - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

6.120.2.10 `static bool decaf::lang::Boolean::parseBoolean (const std::string & value) [static]`

Parses the String passed and extracts an bool.

Parameters

value The std::string value to parse

Returns

bool value

6.120.2.11 `static std::string decaf::lang::Boolean::toString (bool value) [static]`

Converts the bool to a String representation.

Parameters

value The bool value to convert.

Returns

std::string representation of the bool value passed.

6.120.2.12 `std::string decaf::lang::Boolean::toString () const`**Returns**

the string representation of this Boolean's value.

6.120.2.13 `static Boolean decaf::lang::Boolean::valueOf (bool value) [static]`**Parameters**

value The bool value to convert to a Boolean (p. 674) instance.

Returns

a Boolean (p. 674) instance of the primitive boolean value

6.120.2.14 `static Boolean decaf::lang::Boolean::valueOf (const std::string & value) [static]`**Parameters**

value The std::string value to convert to a Boolean (p. 674) instance.

Returns

a Boolean (p. 674) instance of the string value

6.120.3 Field Documentation**6.120.3.1** `const Boolean decaf::lang::Boolean::_FALSE [static]`

The Class object representing the primitive false boolean.

6.120.3.2 `const Boolean decaf::lang::Boolean::_TRUE [static]`

The Class object representing the primitive type boolean.

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**Boolean.h**

6.121 activemq::commands::BooleanExpression Class Reference

```
#include <src/main/activemq/commands/BooleanExpression.h>
```

Inheritance diagram for `activemq::commands::BooleanExpression`:

Public Member Functions

- **BooleanExpression** ()
- virtual **~BooleanExpression** ()
- virtual **DataStructure * cloneDataStructure** () const
Clone this obbect and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src AMQCPP_UNUSED)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1372) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent.*

6.121.1 Constructor & Destructor Documentation

- 6.121.1.1** `activemq::commands::BooleanExpression::BooleanExpression ()`
[inline]
- 6.121.1.2** `virtual activemq::commands::BooleanExpression::~~BooleanExpression ()` [inline, virtual]

6.121.2 Member Function Documentation

- 6.121.2.1** `virtual DataStructure* activemq::commands::BooleanExpression::cloneDataStructure ()` const
[inline, virtual]

Clone this obbect and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1373).

6.121.2.2 `virtual void activemq::commands::BooleanExpression::copyDataStructure (const DataStructure *src AMQCPP_UNUSED) [inline, virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Reimplemented from `activemq::commands::BaseDataStructure` (p. 661).

6.121.2.3 `virtual bool activemq::commands::BooleanExpression::equals (const DataStructure * value) const [inline, virtual]`

Compares the `DataStructure` (p. 1372) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Implements `activemq::commands::DataStructure` (p. 1374).

References `activemq::commands::BaseDataStructure::equals()`.

6.121.2.4 `virtual std::string activemq::commands::BooleanExpression::toString () const [inline, virtual]`

Returns a string containing the information for this `DataStructure` (p. 1372) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from `activemq::commands::BaseDataStructure` (p. 662).

References `activemq::commands::BaseDataStructure::toString()`.

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/BooleanExpression.h`

6.122 activemq::wireformat::openwire::utils::BooleanStream Class Reference

Manages the writing and reading of boolean data streams to and from a data source such as a `DataInputStream` or `DataOutputStream`.

```
#include <src/main/activemq/wireformat/openwire/utils/BooleanStream.h>
```

Public Member Functions

- **BooleanStream** ()
- virtual **~BooleanStream** ()
- bool **readBoolean** () throw (decaf::io::IOException)
Read a boolean data element from the internal data buffer.
- void **writeBoolean** (bool value) throw (decaf::io::IOException)
Writes a Boolean value to the internal data buffer.
- void **marshal** (decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)
Marshal the data to a DataOutputStream.
- void **marshal** (std::vector< unsigned char > &dataOut)
Marshal the data to a STL vector of unsigned chars.
- void **unmarshal** (decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)
Unmarshal a Boolean data stream from the Input Stream.
- void **clear** ()
Clears to old position markers, data starts at the beginning.
- int **marshalledSize** ()
Calc the size that data is marshalled to.

6.122.1 Detailed Description

Manages the writing and reading of boolean data streams to and from a data source such as a DataInputStream or DataOutputStream. The booleans are stored as single bits in the stream, with the stream size pre-pended to the stream when the data is marshalled.

The serialized form of the size field can be between 1 and 3 bytes. If the number of used bytes < 64, size=1 byte If the number of used bytes >=64 and < 256 (size of an unsigned byte), size=2 bytes If the number of used bytes >=256, size=3 bytes

The high-order 2 bits (128 and 64) of the first byte of the size field are used to encode the information about the number of bytes in the size field. The only time the first byte will contain a value >=64 is if there are more bytes in the size field. If the first byte < 64, the value of the byte is simply the size value. If the first byte = 0xC0, the following unsigned byte is the size field. If the first byte = 0x80, the following short (two bytes) are the size field.

6.122.2 Constructor & Destructor Documentation

6.122.2.1 `activemq::wireformat::openwire::utils::BooleanStream::BooleanStream ()`

6.122.2.2 `virtual
activemq::wireformat::openwire::utils::BooleanStream::~~BooleanStream ()` [inline, virtual]

6.122.3 Member Function Documentation

6.122.3.1 `void activemq::wireformat::openwire::utils::BooleanStream::clear ()`

Clears to old position markers, data starts at the beginning.

6.122.3.2 `void activemq::wireformat::openwire::utils::BooleanStream::marshal (std::vector< unsigned char > & dataOut)`

Marshal the data to a STL vector of unsigned chars.

Parameters

dataOut - reference to a vector to write the data to.

6.122.3.3 `void activemq::wireformat::openwire::utils::BooleanStream::marshal (decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)`

Marshal the data to a DataOutputStream.

Parameters

dataOut - Stream to write the data to.

6.122.3.4 `int activemq::wireformat::openwire::utils::BooleanStream::marshalledSize ()`

Calc the size that data is marshalled to.

Returns

int size of marshalled data.

6.122.3.5 `bool activemq::wireformat::openwire::utils::BooleanStream::readBoolean () throw (decaf::io::IOException)`

Read a boolean data element from the internal data buffer.

Returns

boolean from the stream

6.122.3.6 `void activemq::wireformat::openwire::utils::BooleanStream::unmarshal (decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)`

Unmarshal a Boolean data stream from the Input Stream.

Parameters

dataIn - Input Stream to read data from.

6.122.3.7 `void activemq::wireformat::openwire::utils::BooleanStream::writeBoolean (bool value) throw (decaf::io::IOException)`

Writes a Boolean value to the internal data buffer.

Parameters

value - boolean data to write.

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/utils/BooleanStream.h`

6.123 decaf::util::concurrent::BrokenBarrierException Class Reference

```
#include <src/main/decaf/util/concurrent/BrokenBarrierException.h>
```

Inheritance diagram for decaf::util::concurrent::BrokenBarrierException:

Public Member Functions

- **BrokenBarrierException** () throw ()
Default Constructor.
- **BrokenBarrierException** (const decaf::lang::Exception &ex) throw ()
Conversion Constructor from some other Exception.
- **BrokenBarrierException** (const BrokenBarrierException &ex) throw ()
Copy Constructor.
- **BrokenBarrierException** (const std::exception *cause) throw ()
Constructor.
- **BrokenBarrierException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.

- **BrokenBarrierException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **BrokenBarrierException** * clone () const
Clones this exception.
- virtual ~**BrokenBarrierException** () throw ()

6.123.1 Constructor & Destructor Documentation

6.123.1.1 decaf::util::concurrent::BrokenBarrierException::BrokenBarrierException () throw () [inline]

Default Constructor.

6.123.1.2 decaf::util::concurrent::BrokenBarrierException::BrokenBarrierException (const decaf::lang::Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

Parameters

ex An exception that should become this type of Exception

References decaf::lang::Exception::Exception().

6.123.1.3 decaf::util::concurrent::BrokenBarrierException::BrokenBarrierException (const BrokenBarrierException & ex) throw () [inline]

Copy Constructor.

Parameters

ex The Exception to copy in this new instance.

References decaf::lang::Exception::Exception().

6.123.1.4 decaf::util::concurrent::BrokenBarrierException::BrokenBarrierException (const std::exception * cause) throw () [inline]

Constructor.

Parameters

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.123.1.5 `decaf::util::concurrent::BrokenBarrierException::BrokenBarrierException`
 (`const char * file`, `const int lineNumber`, `const char * msg`, ...)
 throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file - The file name where exception occurs
lineNumber - The line number where the exception occurred.
msg - The message to report
 ... - list of primitives that are formatted into the message

6.123.1.6 `decaf::util::concurrent::BrokenBarrierException::BrokenBarrierException`
 (`const char * file`, `const int lineNumber`, `const std::exception * cause`,
`const char * msg`, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file - The file name where exception occurs
lineNumber - The line number where the exception occurred.
cause - The exception that was the cause for this one to be thrown.
msg - The message to report
 ... - list of primitives that are formatted into the message

6.123.1.7 `virtual`
`decaf::util::concurrent::BrokenBarrierException::~~BrokenBarrierException`
 () throw () [inline, virtual]

6.123.2 Member Function Documentation

6.123.2.1 `virtual BrokenBarrierException* decaf::util::concurrent::BrokenBarrierException::clone` ()
 const [inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

a new instance of an exception that is a clone of this one.

Reimplemented from `decaf::lang::Exception` (p. 1479).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/BrokenBarrierException.h`

6.124 activemq::commands::BrokerError Class Reference

This class represents an Exception sent from the Broker.

```
#include <src/main/activemq/commands/BrokerError.h>
```

Inheritance diagram for activemq::commands::BrokerError:

Data Structures

- struct **StackTraceElement**

Public Member Functions

- **BrokerError** ()
- virtual **~BrokerError** ()
- virtual unsigned char **getDataStructureType** () const
*Get the **DataStructure** (p. 1372) Type as defined in CommandTypes.h.*
- virtual **BrokerError** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual **decaf::lang::Pointer**< **commands::Command** > **visit** (**activemq::state::CommandVisitor** *visitor) throw (**exceptions::ActiveMQException**)
Allows a Visitor to visit this command and return a response to the command based on the command type being visited.
- virtual const std::string & **getMessage** () const
Gets the string holding the error message.
- virtual void **setMessage** (const std::string &message)
*Sets the string that contains the error **Message** (p. 2018).*
- virtual const std::string & **getExceptionClass** () const
Gets the string holding the Exception Class name.
- virtual void **setExceptionClass** (const std::string &exceptionClass)
Sets the string that contains the Exception Class name.
- virtual const **decaf::lang::Pointer**< **BrokerError** > & **getCause** () const
Gets the Broker Error that caused this exception.
- virtual void **setCause** (const **decaf::lang::Pointer**< **BrokerError** > &cause)
Sets the Broker Error that caused this exception.

- virtual const std::vector< **decaf::lang::Pointer< StackTraceElement > >** & **getStackTraceElements** () const

Gets the Stack Trace Elements for the Exception.

- virtual void **setStackTraceElements** (const std::vector< **decaf::lang::Pointer< StackTraceElement > >** &stackTraceElements)

Sets the Stack Trace Elements for this Exception.

6.124.1 Detailed Description

This class represents an Exception sent from the Broker. The Broker sends java Throwables, so we must mimic its structure here.

6.124.2 Constructor & Destructor Documentation

6.124.2.1 **activemq::commands::BrokerError::BrokerError** () [inline]

6.124.2.2 **virtual activemq::commands::BrokerError::~~BrokerError** () [virtual]

6.124.3 Member Function Documentation

6.124.3.1 **virtual BrokerError* activemq::commands::BrokerError::cloneDataStructure** () const [inline, virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1373).

References **copyDataStructure()**.

6.124.3.2 **virtual void activemq::commands::BrokerError::copyDataStructure** (const **DataStructure * src**) [virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 598).

Referenced by **cloneDataStructure()**.

6.124.3.3 `virtual const decaf::lang::Pointer<BrokerError>&
activemq::commands::BrokerError::getCause () const [inline,
virtual]`

Gets the Broker Error that caused this exception.

Returns

Broker Error Pointer

6.124.3.4 `virtual unsigned char ac-
tivemq::commands::BrokerError::getDataStructureType () const [inline, virtual]`

Get the **DataStructure** (p. 1372) Type as defined in CommandTypes.h.

Returns

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 1375).

6.124.3.5 `virtual const std::string& ac-
tivemq::commands::BrokerError::getExceptionClass ()
const [inline, virtual]`

Gets the string holding the Exception Class name.

Returns

Exception Class name

6.124.3.6 `virtual const std::string& activemq::commands::BrokerError::getMessage
() const [inline, virtual]`

Gets the string holding the error message.

Returns

String **Message** (p. 2018)

6.124.3.7 `virtual const std::vector< decaf::lang::Pointer<StackTraceElement> >&
activemq::commands::BrokerError::getStackTraceElements () const
[inline, virtual]`

Gets the Stack Trace Elements for the Exception.

Returns

Stack Trace Elements

6.124.3.8 `virtual void activemq::commands::BrokerError::setCause (const decaf::lang::Pointer< BrokerError > & cause) [inline, virtual]`

Sets the Broker Error that caused this exception.

Parameters

cause - Broker Error

6.124.3.9 `virtual void activemq::commands::BrokerError::setExceptionClass (const std::string & exceptionClass) [inline, virtual]`

Sets the string that contains the Exception Class name.

Parameters

exceptionClass - String Exception Class name

6.124.3.10 `virtual void activemq::commands::BrokerError::setMessage (const std::string & message) [inline, virtual]`

Sets the string that contains the error **Message** (p.2018).

Parameters

message - String Error **Message** (p.2018)

6.124.3.11 `virtual void activemq::commands::BrokerError::setStackTraceElements (const std::vector< decaf::lang::Pointer< StackTraceElement > > & stackTraceElements) [inline, virtual]`

Sets the Stack Trace Elements for this Exception.

Parameters

stackTraceElements - Stack Trace Elements

6.124.3.12 `virtual decaf::lang::Pointer<commands::Command> activemq::commands::BrokerError::visit (activemq::state::CommandVisitor * visitor) throw (exceptions::ActiveMQException) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

Returns

a **Response** (p.2611) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 995).

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/BrokerError.h`

6.125 activemq::exceptions::BrokerException Class Reference

```
#include <src/main/activemq/exceptions/BrokerException.h>
```

Inheritance diagram for `activemq::exceptions::BrokerException`:

Public Member Functions

- **BrokerException** () throw ()
- **BrokerException** (const **exceptions::ActiveMQException** &ex) throw ()
- **BrokerException** (const **BrokerException** &ex) throw ()
- **BrokerException** (const char *file, const int lineNumber, const char *msg,...) throw ()
- **BrokerException** (const char *file, const int lineNumber, const **commands::BrokerError** *error) throw ()
- virtual **BrokerException** * **clone** () const
Clones this exception.
- virtual ~**BrokerException** () throw ()

6.125.1 Constructor & Destructor Documentation

6.125.1.1 `activemq::exceptions::BrokerException::BrokerException () throw ()`
[inline]

6.125.1.2 `activemq::exceptions::BrokerException::BrokerException (const exceptions::ActiveMQException & ex) throw ()` [inline]

6.125.1.3 `activemq::exceptions::BrokerException::BrokerException (const BrokerException & ex) throw ()` [inline]

6.125.1.4 `activemq::exceptions::BrokerException::BrokerException (const char * file, const int lineNumber, const char * msg, ...) throw ()`
[inline]

6.125.1.5 `activemq::exceptions::BrokerException::BrokerException (const char * file, const int lineNumber, const commands::BrokerError * error) throw ()` [inline]

References `decaf::lang::Exception::setMark()`, and `decaf::lang::Exception::setMessage()`.

6.125.1.6 `virtual activemq::exceptions::BrokerException::~~BrokerException ()
throw () [inline, virtual]`

6.125.2 Member Function Documentation

6.125.2.1 `virtual BrokerException* activemq::exceptions::BrokerException::clone () const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Reimplemented from `activemq::exceptions::ActiveMQException` (p. 271).

The documentation for this class was generated from the following file:

- `src/main/activemq/exceptions/BrokerException.h`

6.126 `activemq::commands::BrokerId` Class Reference

```
#include <src/main/activemq/commands/BrokerId.h>
```

Inheritance diagram for `activemq::commands::BrokerId`:

Public Types

- `typedef decaf::lang::PointerComparator< BrokerId > COMPARATOR`

Public Member Functions

- `BrokerId ()`
- `BrokerId (const BrokerId &other)`
- `virtual ~BrokerId ()`
- `virtual unsigned char getDataStructureType () const`
Get the unique identifier that this object and its own Marshaler share.
- `virtual BrokerId * cloneDataStructure () const`
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- `virtual void copyDataStructure (const DataStructure *src)`
Copy the contents of the passed object into this object's members, overwriting any existing data.
- `virtual std::string toString () const`
*Returns a string containing the information for this **DataStructure** (p. 1372) such as its type and value of its elements.*
- `virtual bool equals (const DataStructure *value) const`
*Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent.*

- virtual const std::string & **getValue** () const
- virtual std::string & **getValue** ()
- virtual void **setValue** (const std::string &value)
- virtual int **compareTo** (const **BrokerId** &value) const
- virtual bool **equals** (const **BrokerId** &value) const
- virtual bool **operator==** (const **BrokerId** &value) const
- virtual bool **operator<** (const **BrokerId** &value) const
- **BrokerId** & **operator=** (const **BrokerId** &other)

Static Public Attributes

- static const unsigned char **ID_BROKERID** = 124

Protected Attributes

- std::string **value**

6.126.1 Member Typedef Documentation

- 6.126.1.1** typedef decaf::lang::PointerComparator<BrokerId>
 activemq::commands::BrokerId::COMPARATOR

6.126.2 Constructor & Destructor Documentation

- 6.126.2.1** activemq::commands::BrokerId::BrokerId ()
- 6.126.2.2** activemq::commands::BrokerId::BrokerId (const BrokerId & *other*)
- 6.126.2.3** virtual activemq::commands::BrokerId::~~BrokerId () [virtual]

6.126.3 Member Function Documentation

- 6.126.3.1** virtual BrokerId* activemq::commands::BrokerId::cloneDataStructure () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

- 6.126.3.2** virtual int activemq::commands::BrokerId::compareTo (const BrokerId & *value*) const [virtual]
- 6.126.3.3** virtual void activemq::commands::BrokerId::copyDataStructure (const DataStructure * *src*) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

6.126.3.4 `virtual bool activemq::commands::BrokerId::equals (const
DataStructure * value) const` [virtual]

Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

6.126.3.5 `virtual bool activemq::commands::BrokerId::equals (const BrokerId &
value) const` [virtual]

6.126.3.6 `virtual unsigned char ac-
tivemq::commands::BrokerId::getDataStructureType () const` [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataStructure** (p. 1372) type copy.

6.126.3.7 `virtual const std::string& activemq::commands::BrokerId::getValue ()
const` [virtual]

6.126.3.8 `virtual std::string& activemq::commands::BrokerId::getValue ()
[virtual]`

6.126.3.9 `virtual bool activemq::commands::BrokerId::operator< (const BrokerId
& value) const` [virtual]

6.126.3.10 `BrokerId& activemq::commands::BrokerId::operator= (const BrokerId
& other)`

6.126.3.11 `virtual bool activemq::commands::BrokerId::operator== (const
BrokerId & value) const` [virtual]

6.126.3.12 `virtual void activemq::commands::BrokerId::setValue (const std::string
& value)` [virtual]

6.126.3.13 `virtual std::string activemq::commands::BrokerId::toString () const
[virtual]`

Returns a string containing the information for this **DataStructure** (p. 1372) such as its type and value of its elements.

Returns

formatted string useful for debugging.

6.126.4 Field Documentation

6.126.4.1 `const unsigned char activemq::commands::BrokerId::ID_BROKERID = 124 [static]`

6.126.4.2 `std::string activemq::commands::BrokerId::value [protected]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/BrokerId.h`

6.127 activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller Class Reference

Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 694).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/BrokerIdMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller`:

Public Member Functions

- **BrokerIdMarshaller** ()
- virtual **~BrokerIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

6.127.1 Detailed Description

Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 694). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.127.2 Constructor & Destructor Documentation

6.127.2.1 **activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller::BrokerIdMarshaller**
() [inline]

6.127.2.2 **virtual**
activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller::~~BrokerIdMarshaller
() [inline, virtual]

6.127.3 Member Function Documentation

6.127.3.1 **virtual commands::DataStructure*** **activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller::createObject**
() const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1331).

6.127.3.2 **virtual unsigned char** **activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller::getDataStructureType**
() const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1336).

6.127.3.3 virtual void activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1342).

6.127.3.4 virtual void activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1348).

6.127.3.5 virtual int activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1354).

```
6.127.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1360).

```
6.127.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1366).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/BrokerIdMarshaller.h`

6.128 `activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller` Class Reference

Marshaling code for Open Wire Format for `BrokerIdMarshaller` (p. 698).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/BrokerIdMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller`:

Public Member Functions

- `BrokerIdMarshaller ()`
- `virtual ~BrokerIdMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaller.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write a object instance to data output stream.
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)`
Write a object instance to data output stream.

6.128.1 Detailed Description

Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 698). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.128.2 Constructor & Destructor Documentation

6.128.2.1 `activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller::BrokerIdMarshaller () [inline]`

6.128.2.2 `virtual activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller::~~BrokerIdMarshaller () [inline, virtual]`

6.128.3 Member Function Documentation

6.128.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.128.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.128.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1342).

6.128.3.4 virtual void `activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller::looseUnmarshal`
(`OpenWireFormat * wireFormat`, `commands::DataStructure * dataStructure`, `decaf::io::DataInputStream * dataIn`) throw (`decaf::io::IOException`) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1348).

6.128.3.5 virtual int `activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller::tightMarshal1`
(`OpenWireFormat * wireFormat`, `commands::DataStructure * dataStructure`, `utils::BooleanStream * bs`) throw (`decaf::io::IOException`) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1354).

6.128.3.6 `virtual void activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1360).

6.128.3.7 `virtual void activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1366).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/BrokerIdMarshaller.h`

6.129 `activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller` Class Reference

Marshaling code for Open Wire Format for `BrokerIdMarshaller` (p. 701).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/BrokerIdMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller:

Public Member Functions

- **BrokerIdMarshaller** ()
- virtual **~BrokerIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.129.1 Detailed Description

Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 701). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.129.2 Constructor & Destructor Documentation

6.129.2.1 `activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller::BrokerIdMarshaller () [inline]`

6.129.2.2 `virtual
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller::~~BrokerIdMarshaller () [inline, virtual]`

6.129.3 Member Function Documentation

6.129.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.129.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.129.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1342).

6.129.3.4 virtual void activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1348).

6.129.3.5 virtual int activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1354).

6.129.3.6 virtual void activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions

- IOException* if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1360).

```
6.129.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1366).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/BrokerIdMarshaller.h`

6.130 **activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller** Class Reference

Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 705).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/BrokerIdMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller**:

Public Member Functions

- **BrokerIdMarshaller** ()
- virtual **~BrokerIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.130.1 Detailed Description

Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 705). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.130.2 Constructor & Destructor Documentation

6.130.2.1 `activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller::BrokerIdMarshaller () [inline]`

6.130.2.2 `virtual activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller::~~BrokerIdMarshaller () [inline, virtual]`

6.130.3 Member Function Documentation

6.130.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.130.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.130.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1342).

6.130.3.4 virtual void activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1348).

6.130.3.5 virtual int activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1354).

6.130.3.6 virtual void activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions

- IOException* if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1360).

```
6.130.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1366).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/BrokerIdMarshaller.h`

6.131 **activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller** Class Reference

Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 709).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/BrokerIdMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller**:

Public Member Functions

- **BrokerIdMarshaller** ()
- virtual **~BrokerIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.131.1 Detailed Description

Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 709). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.131.2 Constructor & Destructor Documentation

6.131.2.1 `activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller::BrokerIdMarshaller () [inline]`

6.131.2.2 `virtual activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller::~~BrokerIdMarshaller () [inline, virtual]`

6.131.3 Member Function Documentation

6.131.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.131.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.131.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1342).

6.131.3.4 virtual void activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1348).

6.131.3.5 virtual int activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1354).

6.131.3.6 virtual void activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1360).

```
6.131.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1366).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**BrokerIdMarshaller.h**

6.132 activemq::commands::BrokerInfo Class Reference

```
#include <src/main/activemq/commands/BrokerInfo.h>
```

Inheritance diagram for **activemq::commands::BrokerInfo**:

Public Member Functions

- **BrokerInfo** ()
- virtual **~BrokerInfo** ()
- virtual unsigned char **getDataStructureType** () const

Get the unique identifier that this object and its own Marshaler share.

- virtual **BrokerInfo** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1372) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **BrokerId** > & **getBrokerId** () const
- virtual **Pointer**< **BrokerId** > & **getBrokerId** ()
- virtual void **setBrokerId** (const **Pointer**< **BrokerId** > &brokerId)
- virtual const std::string & **getBrokerURL** () const
- virtual std::string & **getBrokerURL** ()
- virtual void **setBrokerURL** (const std::string &brokerURL)
- virtual const std::vector< **decaf::lang::Pointer**< **BrokerInfo** > > & **getPeerBrokerInfos** () const
- virtual std::vector< **decaf::lang::Pointer**< **BrokerInfo** > > & **getPeerBrokerInfos** ()
- virtual void **setPeerBrokerInfos** (const std::vector< **decaf::lang::Pointer**< **BrokerInfo** > > &peerBrokerInfos)
- virtual const std::string & **getBrokerName** () const
- virtual std::string & **getBrokerName** ()
- virtual void **setBrokerName** (const std::string &brokerName)
- virtual bool **isSlaveBroker** () const
- virtual void **setSlaveBroker** (bool slaveBroker)
- virtual bool **isMasterBroker** () const
- virtual void **setMasterBroker** (bool masterBroker)
- virtual bool **isFaultTolerantConfiguration** () const
- virtual void **setFaultTolerantConfiguration** (bool faultTolerantConfiguration)
- virtual bool **isDuplexConnection** () const
- virtual void **setDuplexConnection** (bool duplexConnection)
- virtual bool **isNetworkConnection** () const
- virtual void **setNetworkConnection** (bool networkConnection)
- virtual long long **getConnectionId** () const
- virtual void **setConnectionId** (long long connectionId)
- virtual const std::string & **getBrokerUploadUrl** () const
- virtual std::string & **getBrokerUploadUrl** ()
- virtual void **setBrokerUploadUrl** (const std::string &brokerUploadUrl)
- virtual const std::string & **getNetworkProperties** () const
- virtual std::string & **getNetworkProperties** ()
- virtual void **setNetworkProperties** (const std::string &networkProperties)
- virtual bool **isBrokerInfo** () const
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor *visitor)
throw (exceptions::ActiveMQException)

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_BROKERINFO** = 2

Protected Member Functions

- **BrokerInfo** (const **BrokerInfo** &)
- **BrokerInfo** & operator= (const **BrokerInfo** &)

Protected Attributes

- **Pointer< BrokerId > brokerId**
- std::string **brokerURL**
- std::vector< decaf::lang::Pointer< **BrokerInfo** > > **peerBrokerInfos**
- std::string **brokerName**
- bool **slaveBroker**
- bool **masterBroker**
- bool **faultTolerantConfiguration**
- bool **duplexConnection**
- bool **networkConnection**
- long long **connectionId**
- std::string **brokerUploadUrl**
- std::string **networkProperties**

6.132.1 Constructor & Destructor Documentation

6.132.1.1 `activemq::commands::BrokerInfo::BrokerInfo (const BrokerInfo &)`
[inline, protected]

6.132.1.2 `activemq::commands::BrokerInfo::BrokerInfo ()`

6.132.1.3 `virtual activemq::commands::BrokerInfo::~~BrokerInfo ()` [virtual]

6.132.2 Member Function Documentation

6.132.2.1 `virtual BrokerInfo* activemq::commands::BrokerInfo::cloneDataStructure () const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1373).

6.132.2.2 virtual void activemq::commands::BrokerInfo::copyDataStructure (const DataStructure * *src*) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 598).

6.132.2.3 virtual bool activemq::commands::BrokerInfo::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 598).

- 6.132.2.4 `virtual Pointer<BrokerId>& activemq::commands::BrokerInfo::getBrokerId ()`
[virtual]
- 6.132.2.5 `virtual const Pointer<BrokerId>& activemq::commands::BrokerInfo::getBrokerId () const`
[virtual]
- 6.132.2.6 `virtual const std::string& activemq::commands::BrokerInfo::getBrokerName ()`
const [virtual]
- 6.132.2.7 `virtual std::string& activemq::commands::BrokerInfo::getBrokerName ()`
[virtual]
- 6.132.2.8 `virtual std::string& activemq::commands::BrokerInfo::getBrokerUploadUrl ()`
[virtual]
- 6.132.2.9 `virtual const std::string& activemq::commands::BrokerInfo::getBrokerUploadUrl ()`
const [virtual]
- 6.132.2.10 `virtual const std::string& activemq::commands::BrokerInfo::getBrokerURL ()`
const [virtual]
- 6.132.2.11 `virtual std::string& activemq::commands::BrokerInfo::getBrokerURL ()`
[virtual]
- 6.132.2.12 `virtual long long activemq::commands::BrokerInfo::getConnectionId () const`
[virtual]
- 6.132.2.13 `virtual unsigned char activemq::commands::BrokerInfo::getDataStructureType () const`
[virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataStructure** (p. 1372) type copy.

Implements **activemq::commands::DataStructure** (p. 1375).

- 6.132.2.14 `virtual const std::string& activemq::commands::BrokerInfo::getNetworkProperties () const [virtual]`
- 6.132.2.15 `virtual std::string& activemq::commands::BrokerInfo::getNetworkProperties () [virtual]`
- 6.132.2.16 `virtual const std::vector< decaf::lang::Pointer<BrokerInfo> >& activemq::commands::BrokerInfo::getPeerBrokerInfos () const [virtual]`
- 6.132.2.17 `virtual std::vector< decaf::lang::Pointer<BrokerInfo> >& activemq::commands::BrokerInfo::getPeerBrokerInfos () [virtual]`
- 6.132.2.18 `virtual bool activemq::commands::BrokerInfo::isBrokerInfo () const [inline, virtual]`

Returns

an answer of true to the `isBrokerInfo()` (p. 718) query.

Reimplemented from `activemq::commands::BaseCommand` (p. 600).

- 6.132.2.19 virtual bool activemq::commands::BrokerInfo::isDuplexConnection () const [virtual]
- 6.132.2.20 virtual bool activemq::commands::BrokerInfo::isFaultTolerantConfiguration () const [virtual]
- 6.132.2.21 virtual bool activemq::commands::BrokerInfo::isMasterBroker () const [virtual]
- 6.132.2.22 virtual bool activemq::commands::BrokerInfo::isNetworkConnection () const [virtual]
- 6.132.2.23 virtual bool activemq::commands::BrokerInfo::isSlaveBroker () const [virtual]
- 6.132.2.24 BrokerInfo& activemq::commands::BrokerInfo::operator= (const BrokerInfo &) [inline, protected]
- 6.132.2.25 virtual void activemq::commands::BrokerInfo::setBrokerId (const Pointer< BrokerId > & *brokerId*) [virtual]
- 6.132.2.26 virtual void activemq::commands::BrokerInfo::setBrokerName (const std::string & *brokerName*) [virtual]
- 6.132.2.27 virtual void activemq::commands::BrokerInfo::setBrokerUploadUrl (const std::string & *brokerUploadUrl*) [virtual]
- 6.132.2.28 virtual void activemq::commands::BrokerInfo::setBrokerURL (const std::string & *brokerURL*) [virtual]
- 6.132.2.29 virtual void activemq::commands::BrokerInfo::setConnectionId (long long *connectionId*) [virtual]
- 6.132.2.30 virtual void activemq::commands::BrokerInfo::setDuplexConnection (bool *duplexConnection*) [virtual]
- 6.132.2.31 virtual void activemq::commands::BrokerInfo::setFaultTolerantConfiguration (bool *faultTolerantConfiguration*) [virtual]
- 6.132.2.32 virtual void activemq::commands::BrokerInfo::setMasterBroker (bool *masterBroker*) [virtual]
- 6.132.2.33 virtual void activemq::commands::BrokerInfo::setNetworkConnection (bool *networkConnection*) [virtual]
- 6.132.2.34 virtual void activemq::commands::BrokerInfo::setNetworkProperties (const std::string & *networkProperties*) [virtual]
- 6.132.2.35 virtual void activemq::commands::BrokerInfo::setPeerBrokerInfos (const std::vector< decaf::lang::Pointer< BrokerInfo > > & *peerBrokerInfos*) [virtual]
- 6.132.2.36 virtual void activemq::commands::BrokerInfo::setSlaveBroker (bool *slaveBroker*) [virtual]
- 6.132.2.37 virtual std::string activemq::commands::BrokerInfo::toString () const [virtual]

Returns a string containing the information for this **DataStructure** (p.1372) such as its type and value of its elements

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 602).

6.132.2.38 `virtual Pointer<Command> activemq::commands::BrokerInfo::visit
(activemq::state::CommandVisitor * visitor) throw (
exceptions::ActiveMQException) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

Returns

a **Response** (p. 2611) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 995).

6.132.3 Field Documentation

- 6.132.3.1 `Pointer<BrokerId> activemq::commands::BrokerInfo::brokerId` [protected]
- 6.132.3.2 `std::string activemq::commands::BrokerInfo::brokerName` [protected]
- 6.132.3.3 `std::string activemq::commands::BrokerInfo::brokerUploadUrl` [protected]
- 6.132.3.4 `std::string activemq::commands::BrokerInfo::brokerURL` [protected]
- 6.132.3.5 `long long activemq::commands::BrokerInfo::connectionId` [protected]
- 6.132.3.6 `bool activemq::commands::BrokerInfo::duplexConnection` [protected]
- 6.132.3.7 `bool activemq::commands::BrokerInfo::faultTolerantConfiguration` [protected]
- 6.132.3.8 `const unsigned char activemq::commands::BrokerInfo::ID _ - BROKERINFO = 2` [static]
- 6.132.3.9 `bool activemq::commands::BrokerInfo::masterBroker` [protected]
- 6.132.3.10 `bool activemq::commands::BrokerInfo::networkConnection` [protected]
- 6.132.3.11 `std::string activemq::commands::BrokerInfo::networkProperties` [protected]
- 6.132.3.12 `std::vector< decaf::lang::Pointer<BrokerInfo> > activemq::commands::BrokerInfo::peerBrokerInfos` [protected]
- 6.132.3.13 `bool activemq::commands::BrokerInfo::slaveBroker` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/BrokerInfo.h`

6.133 `activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller` Class Reference

Marshaling code for Open Wire Format for `BrokerInfoMarshaller` (p. 721).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/BrokerInfoMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller`:

Public Member Functions

- `BrokerInfoMarshaller ()`

- virtual `~BrokerInfoMarshaller ()`

- virtual `commands::DataStructure * createObject () const`

Creates a new instance of this marshalable type.

- virtual unsigned char `getDataStructureType () const`

Get the Data Structure Type that identifies this Marshaller.

- virtual void `tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`

Un-marshal an object instance from the data input stream.

- virtual int `tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`

Write the booleans that this object uses to a BooleanStream.

- virtual void `tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)`

Write a object instance to data output stream.

- virtual void `looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)`

Un-marshal an object instance from the data input stream.

- virtual void `looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)`

Write a object instance to data output stream.

6.133.1 Detailed Description

Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 721). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.133.2 Constructor & Destructor Documentation

6.133.2.1 `activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller::BrokerInfoMarshaller () [inline]`

6.133.2.2 `virtual activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller::~~BrokerInfoMarshaller () [inline, virtual]`

6.133.3 Member Function Documentation

6.133.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.133.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.133.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 604).

6.133.3.4 virtual void activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 605).

6.133.3.5 virtual int activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 606).

6.133.3.6 virtual void activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 608).

```
6.133.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 609).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/BrokerInfoMarshaller.h`

6.134 `activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller` Class Reference

Marshaling code for Open Wire Format for `BrokerInfoMarshaller` (p. 725).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/BrokerInfoMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller`:

Public Member Functions

- **BrokerInfoMarshaller** ()
- virtual **~BrokerInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.134.1 Detailed Description

Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 725). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.134.2 Constructor & Destructor Documentation

6.134.2.1 `activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller::BrokerInfoMarshaller()` [inline]

6.134.2.2 `virtual activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller::~~BrokerInfoMarshaller()` [inline, virtual]

6.134.3 Member Function Documentation

6.134.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.134.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.134.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 611).

6.134.3.4 virtual void activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 612).

6.134.3.5 virtual int activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 613).

6.134.3.6 virtual void activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 614).

```
6.134.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 615).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/BrokerInfoMarshaller.h`

6.135 `activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller` Class Reference

Marshaling code for Open Wire Format for `BrokerInfoMarshaller` (p. 729).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/BrokerInfoMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller`:

Public Member Functions

- **BrokerInfoMarshaller** ()
- virtual **~BrokerInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.135.1 Detailed Description

Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 729). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.135.2 Constructor & Destructor Documentation

6.135.2.1 `activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller::BrokerInfoMarshaller () [inline]`

6.135.2.2 `virtual activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller::~~BrokerInfoMarshaller () [inline, virtual]`

6.135.3 Member Function Documentation

6.135.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.135.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.135.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 618).

6.135.3.4 virtual void activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 619).

6.135.3.5 virtual int activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 620).

6.135.3.6 virtual void activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 621).

```
6.135.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 622).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/BrokerInfoMarshaller.h`

6.136 `activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller` Class Reference

Marshaling code for Open Wire Format for `BrokerInfoMarshaller` (p. 733).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/BrokerInfoMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller`:

Public Member Functions

- **BrokerInfoMarshaller** ()
- virtual **~BrokerInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.136.1 Detailed Description

Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 733). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.136.2 Constructor & Destructor Documentation

6.136.2.1 `activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller::BrokerInfoMarshaller () [inline]`

6.136.2.2 `virtual activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller::~~BrokerInfoMarshaller () [inline, virtual]`

6.136.3 Member Function Documentation

6.136.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.136.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.136.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 624).

6.136.3.4 virtual void activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 625).

6.136.3.5 virtual int activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 626).

6.136.3.6 virtual void activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 628).

```
6.136.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 629).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/BrokerInfoMarshaller.h`

6.137 `activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller` Class Reference

Marshaling code for Open Wire Format for `BrokerInfoMarshaller` (p. 737).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/BrokerInfoMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller`:

Public Member Functions

- **BrokerInfoMarshaller** ()
- virtual **~BrokerInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.137.1 Detailed Description

Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 737). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.137.2 Constructor & Destructor Documentation

6.137.2.1 `activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller::BrokerInfoMarshaller()` [inline]

6.137.2.2 `virtual activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller::~~BrokerInfoMarshaller()` [inline, virtual]

6.137.3 Member Function Documentation

6.137.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.137.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.137.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 631).

6.137.3.4 virtual void activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 632).

6.137.3.5 virtual int activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 633).

6.137.3.6 virtual void activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 634).

```
6.137.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 635).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/BrokerInfoMarshaller.h`

6.138 decaf::nio::Buffer Class Reference

A container for data of a specific primitive type.

```
#include <src/main/decaf/nio/Buffer.h>
```

Inheritance diagram for `decaf::nio::Buffer`:

Public Member Functions

- **Buffer** (std::size_t capacity)
- **Buffer** (const **Buffer** &other)
- virtual ~**Buffer** ()
- virtual std::size_t **capacity** () const
- virtual std::size_t **position** () const
- virtual **Buffer** & **position** (std::size_t newPosition) throw (lang::exceptions::IllegalArgumentException)
Sets this buffer's position.
- virtual std::size_t **limit** () const
- virtual **Buffer** & **limit** (std::size_t newLimit) throw (lang::exceptions::IllegalArgumentException)
Sets this buffer's limit.
- virtual **Buffer** & **mark** ()
Sets this buffer's mark at its position.
- virtual **Buffer** & **reset** () throw (InvalidMarkException)
Resets this buffer's position to the previously-marked position.
- virtual **Buffer** & **clear** ()
Clears this buffer.
- virtual **Buffer** & **flip** ()
Flips this buffer.
- virtual **Buffer** & **rewind** ()
Rewinds this buffer.
- virtual std::size_t **remaining** () const
Returns the number of elements between the current position and the limit.
- virtual bool **hasRemaining** () const
Tells whether there are any elements between the current position and the limit.
- virtual bool **isReadOnly** () const =0
Tells whether or not this buffer is read-only.

Protected Attributes

- std::size_t **_position**
- std::size_t **_capacity**
- std::size_t **_limit**
- std::size_t **_mark**
- bool **_markSet**

6.138.1 Detailed Description

A container for data of a specific primitive type. A buffer is a linear, finite sequence of elements of a specific primitive type. Aside from its content, the essential properties of a buffer are its capacity, limit, and position:

A buffer's capacity is the number of elements it contains. The capacity of a buffer is never negative and never changes.

A buffer's limit is the index of the first element that should not be read or written. A buffer's limit is never negative and is never greater than its capacity.

A buffer's position is the index of the next element to be read or written. A buffer's position is never negative and is never greater than its limit.

There is one subclass of this class for each non-boolean primitive type.

Transferring data: Each subclass of this class defines two categories of get and put operations: * Relative operations read or write one or more elements starting at the current position and then increment the position by the number of elements transferred. If the requested transfer exceeds the limit then a relative get operation throws a **BufferUnderflowException** (p. 774) and a relative put operation throws a **BufferOverflowException** (p. 771); in either case, no data is transferred. * Absolute operations take an explicit element index and do not affect the position. Absolute get and put operations throw an **IndexOutOfBoundsException** if the index argument exceeds the limit.

Data may also, of course, be transferred in to or out of a buffer by the I/O operations of an appropriate channel, which are always relative to the current position.

Marking and resetting:

A buffer's mark is the index to which its position will be reset when the reset method is invoked. The mark is not always defined, but when it is defined it is never negative and is never greater than the position. If the mark is defined then it is discarded when the position or the limit is adjusted to a value smaller than the mark. If the mark is not defined then invoking the reset method causes an **InvalidMarkException** (p. 1700) to be thrown.

Invariants:

The following invariant holds for the mark, position, limit, and capacity values: $0 \leq \text{mark} \leq \text{position} \leq \text{limit} \leq \text{capacity}$

A newly-created buffer always has a position of zero and a mark that is undefined. The initial limit may be zero, or it may be some other value that depends upon the type of the buffer and the manner in which it is constructed. The initial content of a buffer is, in general, undefined.

Clearing, flipping, and rewinding:

In addition to methods for accessing the position, limit, and capacity values and for marking and resetting, this class also defines the following operations upon buffers:

clear() (p. 744) makes a buffer ready for a new sequence of channel-read or relative put operations: It sets the limit to the capacity and the position to zero.

flip() (p. 744) makes a buffer ready for a new sequence of channel-write or relative get operations: It sets the limit to the current position and then sets the position to zero.

rewind() (p. 747) makes a buffer ready for re-reading the data that it already contains: It leaves the limit unchanged and sets the position to zero.

Read-only buffers:

Every buffer is readable, but not every buffer is writable. The mutation methods of each buffer

class are specified as optional operations that will throw a **ReadOnlyBufferException** (p. 2520) when invoked upon a read-only buffer. A read-only buffer does not allow its content to be changed, but its mark, position, and limit values are mutable. Whether or not a buffer is read-only may be determined by invoking its `isReadOnly` method.

Thread safety:

Buffers are not safe for use by multiple concurrent threads. If a buffer is to be used by more than one thread then access to the buffer should be controlled by appropriate synchronization.

Invocation chaining:

Methods in this class that do not otherwise have a value to return are specified to return the buffer upon which they are invoked. This allows method invocations to be chained; for example, the sequence of statements

```
b.flip(); b.position(23); b.limit(42);
```

can be replaced by the single, more compact statement `b.flip().position(23).limit(42);`

6.138.2 Constructor & Destructor Documentation

6.138.2.1 `decaf::nio::Buffer::Buffer (std::size_t capacity)`

6.138.2.2 `decaf::nio::Buffer::Buffer (const Buffer & other)`

6.138.2.3 `virtual decaf::nio::Buffer::~~Buffer () [inline, virtual]`

6.138.3 Member Function Documentation

6.138.3.1 `virtual std::size_t decaf::nio::Buffer::capacity () const [inline, virtual]`

Returns

this buffer's capacity.

6.138.3.2 `virtual Buffer& decaf::nio::Buffer::clear () [virtual]`

Clears this buffer.

The position is set to zero, the limit is set to the capacity, and the mark is discarded.

Invoke this method before using a sequence of channel-read or put operations to fill this buffer. For example:

```
buf.clear(); // Prepare buffer for reading in.read(buf); // Read data
```

This method does not actually erase the data in the buffer, but it is named as if it did because it will most often be used in situations in which that might as well be the case.

Returns

a reference to this buffer.

6.138.3.3 `virtual Buffer& decaf::nio::Buffer::flip () [virtual]`

Flips this buffer.

The limit is set to the current position and then the position is set to zero. If the mark is defined then it is discarded.

After a sequence of channel-read or put operations, invoke this method to prepare for a sequence of channel-write or relative get operations. For example:

```
buf.put(magic); // Prepend header
in.read(buf); // Read data into rest of buffer
buf.flip(); // Flip buffer
out.write(buf); // Write header + data to channel
```

This method is often used in conjunction with the compact method when transferring data from one place to another.

Returns

a reference to this buffer.

6.138.3.4 `virtual bool decaf::nio::Buffer::hasRemaining () const [inline, virtual]`

Tells whether there are any elements between the current position and the limit.

Returns

true if, and only if, there is at least one element remaining in this buffer

6.138.3.5 `virtual bool decaf::nio::Buffer::isReadOnly () const [pure virtual]`

Tells whether or not this buffer is read-only.

Returns

true if, and only if, this buffer is read-only

6.138.3.6 `virtual std::size_t decaf::nio::Buffer::limit () const [inline, virtual]`

Returns

this buffers Limit

6.138.3.7 `virtual Buffer& decaf::nio::Buffer::limit (std::size_t newLimit) throw (lang::exceptions::IllegalArgumentException) [virtual]`

Sets this buffer's limit.

If the position is larger than the new limit then it is set to the new limit. If the mark is defined and larger than the new limit then it is discarded.

Parameters

newLimit - The new limit value; must be no larger than this buffer's capacity

Returns

A reference to This buffer

Exceptions

IllegalArgumentException if preconditions on the new pos don't hold.

6.138.3.8 virtual Buffer& decaf::nio::Buffer::mark () [virtual]

Sets this buffer's mark at its position.

Returns

a reference to this buffer.

6.138.3.9 virtual std::size_t decaf::nio::Buffer::position () const [inline, virtual]

Returns

the current position in the buffer

6.138.3.10 virtual Buffer& decaf::nio::Buffer::position (std::size_t newPosition) throw (lang::exceptions::IllegalArgumentException) [virtual]

Sets this buffer's position.

If the mark is defined and larger than the new position then it is discarded.

Parameters

newPosition - the new position in the buffer to set.

Returns

A reference to This buffer

Exceptions

IllegalArgumentException if preconditions on the new pos don't hold.

6.138.3.11 virtual std::size_t decaf::nio::Buffer::remaining () const [inline, virtual]

Returns the number of elements between the current position and the limit.

Returns

The number of elements remaining in this buffer

6.138.3.12 `virtual Buffer& decaf::nio::Buffer::reset () throw (InvalidMarkException) [virtual]`

Resets this buffer's position to the previously-marked position.

Returns

a reference to this buffer.

Exceptions

InvalidMarkException (p. 1700) - If the mark has not been set

6.138.3.13 `virtual Buffer& decaf::nio::Buffer::rewind () [virtual]`

Rewinds this buffer.

The position is set to zero and the mark is discarded.

Invoke this method before a sequence of channel-write or get operations, assuming that the limit has already been set appropriately. For example:

```
out.write(buf); // Write remaining data buf.rewind(); // Rewind buffer buf.get(array); // Copy data into array
```

Returns

a reference to this buffer.

6.138.4 Field Documentation

6.138.4.1 `std::size_t decaf::nio::Buffer::_capacity [protected]`

6.138.4.2 `std::size_t decaf::nio::Buffer::_limit [protected]`

6.138.4.3 `std::size_t decaf::nio::Buffer::_mark [protected]`

6.138.4.4 `bool decaf::nio::Buffer::_markSet [protected]`

6.138.4.5 `std::size_t decaf::nio::Buffer::_position [mutable, protected]`

The documentation for this class was generated from the following file:

- `src/main/decaf/nio/Buffer.h`

6.139 `decaf::io::BufferedInputStream` Class Reference

A wrapper around another input stream that performs a buffered read, where it reads more data than it needs in order to reduce the number of io operations on the input stream.

```
#include <src/main/decaf/io/BufferedInputStream.h>
```

Inheritance diagram for `decaf::io::BufferedInputStream`:

Public Member Functions

- **BufferedInputStream** (**InputStream** *stream, bool **own**=false)
Constructor.
- **BufferedInputStream** (**InputStream** *stream, std::size_t bufferSize, bool **own**=false) throw (lang::exceptions::IllegalArgumentException)
Constructor.
- virtual ~**BufferedInputStream** ()
- virtual std::size_t **available** () const throw (IOException)
Indicates the number of bytes available.
- virtual void **close** () throw (IOException)
*Close this **BufferedInputStream** (p. 747).*
- virtual int **read** () throw (IOException)
Reads a single byte from the buffer.
- virtual int **read** (unsigned char *buffer, std::size_t offset, std::size_t bufferSize) throw (IOException, lang::exceptions::NullPointerException)
Reads an array of bytes from the buffer.
- virtual std::size_t **skip** (std::size_t num) throw (io::IOException, lang::exceptions::UnsupportedOperationException)
Skips over and discards n bytes of data from this input stream.
- virtual void **mark** (int readLimit)
Marks the current position in the stream A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.
- virtual void **reset** () throw (IOException)
Repositions this stream to the position at the time the mark method was last called on this input stream.
- virtual bool **markSupported** () const
Determines if this input stream supports the mark and reset methods.

6.139.1 Detailed Description

A wrapper around another input stream that performs a buffered read, where it reads more data than it needs in order to reduce the number of io operations on the input stream.

6.139.2 Constructor & Destructor Documentation

6.139.2.1 decaf::io::BufferedInputStream::BufferedInputStream (**InputStream** *stream, bool *own* = false)

Constructor.

Parameters

stream The target input stream.

own indicates if we own the stream object, defaults to false

6.139.2.2 `decaf::io::BufferedInputStream::BufferedInputStream (InputStream
* stream, std::size_t bufferSize, bool own = false) throw (
lang::exceptions::IllegalArgumentException)`

Constructor.

Parameters

stream the target input stream

bufferSize the size for the internal buffer.

own indicates if we own the stream object, defaults to false.

Exceptions

IllegalArgumentException is the size is zero.

6.139.2.3 `virtual decaf::io::BufferedInputStream::~~BufferedInputStream ()
[virtual]`

6.139.3 Member Function Documentation

6.139.3.1 `virtual std::size_t decaf::io::BufferedInputStream::available () const
throw (IOException) [inline, virtual]`

Indicates the number of bytes available.

Returns

the sum of the amount of data available in the buffer and the data available on the target input stream.

Reimplemented from `decaf::io::FilterInputStream` (p. 1530).

6.139.3.2 `virtual void decaf::io::BufferedInputStream::close () throw (
IOException) [virtual]`

Close this `BufferedInputStream` (p. 747).

This implementation closes the target stream and releases any resources associated with it.

Exceptions

IOException (p. 1707) If an error occurs attempting to close this stream.

Reimplemented from `decaf::io::FilterInputStream` (p. 1530).

6.139.3.3 virtual void decaf::io::BufferedInputStream::mark (int readLimit) [inline, virtual]

Marks the current position in the stream. A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.

If a stream instance reports that marks are supported then the stream will ensure that the same bytes can be read again after the reset method is called so long the readLimit is not reached.

Parameters

readLimit max bytes read before marked position is invalid.

Reimplemented from **decaf::io::FilterInputStream** (p. 1531).

6.139.3.4 virtual bool decaf::io::BufferedInputStream::markSupported () const [inline, virtual]

Determines if this input stream supports the mark and reset methods.

Whether or not mark and reset are supported is an invariant property of a particular input stream instance.

Returns

true if this stream instance supports marks

Reimplemented from **decaf::io::FilterInputStream** (p. 1531).

6.139.3.5 virtual int decaf::io::BufferedInputStream::read (unsigned char * *buffer*, std::size_t *offset*, std::size_t *bufferSize*) throw (IOException, lang::exceptions::NullPointerException) [virtual]

Reads an array of bytes from the buffer.

Blocks until the requested number of bytes are available.

Parameters

buffer (out) the target buffer.

offset the position in the buffer to start reading from.

bufferSize the size of the output buffer.

Returns

The number of bytes read or -1 if EOF

Exceptions

IOException (p. 1707) thrown if an error occurs.

NullPointerException if buffer is NULL

Reimplemented from **decaf::io::FilterInputStream** (p. 1533).

6.139.3.6 `virtual int decaf::io::BufferedInputStream::read () throw (IOException) [virtual]`

Reads a single byte from the buffer.

Blocks until the data is available.

Returns

The next byte.

Exceptions

IOException (p. 1707) thrown if an error occurs.

Reimplemented from `decaf::io::FilterInputStream` (p. 1532).

6.139.3.7 `virtual void decaf::io::BufferedInputStream::reset () throw (IOException) [inline, virtual]`

Repositions this stream to the position at the time the mark method was last called on this input stream.

If the method `markSupported` returns true, then: * If the method `mark` has not been called since the stream was created, or the number of bytes read from the stream since `mark` was last called is larger than the argument to `mark` at that last call, then an **IOException** (p. 1707) might be thrown. * If such an **IOException** (p. 1707) is not thrown, then the stream is reset to a state such that all the bytes read since the most recent call to `mark` (or since the start of the file, if `mark` has not been called) will be resupplied to subsequent callers of the `read` method, followed by any bytes that otherwise would have been the next input data as of the time of the call to `reset`. If the method `markSupported` returns false, then: * The call to `reset` may throw an **IOException** (p. 1707). * If an **IOException** (p. 1707) is not thrown, then the stream is reset to a fixed state that depends on the particular type of the input stream and how it was created. The bytes that will be supplied to subsequent callers of the `read` method depend on the particular type of the input stream.

Exceptions

IOException (p. 1707)

Reimplemented from `decaf::io::FilterInputStream` (p. 1533).

6.139.3.8 `virtual std::size_t decaf::io::BufferedInputStream::skip (std::size_t num) throw (io::IOException, lang::exceptions::UnsupportedOperationException) [virtual]`

Skips over and discards n bytes of data from this input stream.

The `skip` method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before n bytes have been skipped is only one possibility. The actual number of bytes skipped is returned. If n is negative, no bytes are skipped.

The `skip` method of **InputStream** (p. 1630) creates a byte array and then repeatedly reads into it until n bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.

Parameters

num - the number of bytes to skip

Returns

total butes skipped

Exceptions

IOException (p. 1707) if an error occurs

Reimplemented from **decaf::io::FilterInputStream** (p. 1534).

The documentation for this class was generated from the following file:

- `src/main/decaf/io/BufferedInputStream.h`

6.140 decaf::io::BufferedOutputStream Class Reference

Wrapper around another output stream that buffers output before writing to the target output stream.

```
#include <src/main/decaf/io/BufferedOutputStream.h>
```

Inheritance diagram for decaf::io::BufferedOutputStream:

Public Member Functions

- **BufferedOutputStream** (**OutputStream** *stream, bool **own**=false)
Constructor.
- **BufferedOutputStream** (**OutputStream** *stream, std::size_t bufferSize, bool **own**=false) throw (lang::exceptions::IllegalArgumentException)
Constructor.
- virtual ~**BufferedOutputStream** ()
- virtual void **write** (unsigned char c) throw (IOException)
Writes a single byte to the output stream.
- virtual void **write** (const std::vector< unsigned char > &buffer) throw (IOException)
Writes an array of bytes to the output stream.
- virtual void **write** (const unsigned char *buffer, std::size_t offset, std::size_t len) throw (IOException, lang::exceptions::NullPointerException)
Writes an array of bytes to the output stream.
- virtual void **flush** () throw (IOException)
Invokes flush on the target output stream.
- void **close** () throw (io::IOException)
Invokes close on the target output stream.

6.140.1 Detailed Description

Wrapper around another output stream that buffers output before writing to the target output stream.

6.140.2 Constructor & Destructor Documentation

6.140.2.1 `decaf::io::BufferedOutputStream::BufferedOutputStream (OutputStream * stream, bool own = false)`

Constructor.

Parameters

- stream* The target output stream.
- own* Indicates if this class owns the stream pointer.

6.140.2.2 `decaf::io::BufferedOutputStream::BufferedOutputStream (OutputStream * stream, std::size_t bufferSize, bool own = false) throw (lang::exceptions::IllegalArgumentException)`

Constructor.

Parameters

- stream* The target output stream.
- bufferSize* The size for the internal buffer.
- own* Indicates if this class owns the stream pointer.

6.140.2.3 `virtual decaf::io::BufferedOutputStream::~~BufferedOutputStream ()` [virtual]

6.140.3 Member Function Documentation

6.140.3.1 `void decaf::io::BufferedOutputStream::close () throw (io::IOException)` [virtual]

Invokes close on the target output stream.

Exceptions

- IOException* (p. 1707) thrown if an error occurs.

Reimplemented from `decaf::io::FilterOutputStream` (p. 1539).

6.140.3.2 `virtual void decaf::io::BufferedOutputStream::flush () throw (IOException)` [virtual]

Invokes flush on the target output stream.

Exceptions

IOException (p. 1707) thrown if an error occurs.

Reimplemented from `decaf::io::FilterOutputStream` (p. 1539).

6.140.3.3 `virtual void decaf::io::BufferedOutputStream::write (const std::vector< unsigned char > & buffer) throw (IOException) [virtual]`

Writes an array of bytes to the output stream.

Parameters

buffer The bytes to write.

Exceptions

IOException (p. 1707) thrown if an error occurs.

Reimplemented from `decaf::io::FilterOutputStream` (p. 1542).

6.140.3.4 `virtual void decaf::io::BufferedOutputStream::write (const unsigned char * buffer, std::size_t offset, std::size_t len) throw (IOException, lang::exceptions::NullPointerException) [virtual]`

Writes an array of bytes to the output stream.

Parameters

buffer The array of bytes to write.

offset the position to start writing in buffer.

len The number of bytes from the buffer to be written.

Exceptions

IOException (p. 1707) thrown if an error occurs.

NullPointerException thrown if buffer is Null.

Reimplemented from `decaf::io::FilterOutputStream` (p. 1543).

6.140.3.5 `virtual void decaf::io::BufferedOutputStream::write (unsigned char c) throw (IOException) [virtual]`

Writes a single byte to the output stream.

Parameters

c the byte.

Exceptions

IOException (p. 1707) thrown if an error occurs.

Reimplemented from **decaf::io::FilterOutputStream** (p. 1542).

The documentation for this class was generated from the following file:

- `src/main/decaf/io/BufferedOutputStream.h`

6.141 decaf::net::BufferedSocket Class Reference

Buffered **Socket** (p. 2786) class that wraps a **Socket** (p. 2786) derived object and provides Buffered input and Output Streams to improve the efficiency of the reads and writes.

```
#include <src/main/decaf/net/BufferedSocket.h>
```

Inheritance diagram for decaf::net::BufferedSocket:

Public Member Functions

- **BufferedSocket** (**Socket** *socket, int inputBufferSize=1000, int outputBufferSize=1000, bool own=true)
Constructs a new Buffered socket object.
- virtual **~BufferedSocket** ()
- virtual void **connect** (const char *host, int port) throw (SocketException)
Connects to the specified destination.
- virtual void **close** () throw (decaf::io::IOException)
Closes this object and deallocates the appropriate resources.
- virtual bool **isConnected** () const
Indicates whether or not this socket is connected to a destination.
- virtual **io::InputStream** * **getInputStream** ()
Gets the InputStream for this socket.
- virtual **io::OutputStream** * **getOutputStream** ()
Gets the OutputStream for this socket.
- virtual int **getSoLinger** () const throw (SocketException)
Gets the linger time.
- virtual void **setSoLinger** (int linger) throw (SocketException)
Sets the linger time.
- virtual bool **getKeepAlive** () const throw (SocketException)
Gets the keep alive flag.
- virtual void **setKeepAlive** (bool keepAlive) throw (SocketException)
Enables/disables the keep alive flag.

- virtual int **getReceiveBufferSize** () const throw (SocketException)
Gets the receive buffer size.
- virtual void **setReceiveBufferSize** (int size) throw (SocketException)
Sets the receive buffer size.
- virtual bool **getReuseAddress** () const throw (SocketException)
Gets the reuse address flag.
- virtual void **setReuseAddress** (bool reuse) throw (SocketException)
Sets the reuse address flag.
- virtual int **getSendBufferSize** () const throw (SocketException)
Gets the send buffer size.
- virtual void **setSendBufferSize** (int size) throw (SocketException)
Sets the send buffer size.
- virtual int **getSoTimeout** () const throw (SocketException)
Gets the timeout for socket operations.
- virtual void **setSoTimeout** (int timeout) throw (SocketException)
Sets the timeout for socket operations.

6.141.1 Detailed Description

Buffered **Socket** (p.2786) class that wraps a **Socket** (p.2786) derived object and provides Buffered input and Output Streams to improve the efficiency of the reads and writes.

6.141.2 Constructor & Destructor Documentation

6.141.2.1 decaf::net::BufferedSocket::BufferedSocket (Socket * *socket*, int *inputBufferSize* = 1000, int *outputBufferSize* = 1000, bool *own* = true)

Constructs a new Buffered socket object.

Parameters

socket the socket to buffer

inputBufferSize size of the input buffer

outputBufferSize size of the output buffer

own does this object own the passed socket

6.141.2.2 virtual decaf::net::BufferedSocket::~~BufferedSocket () [virtual]

6.141.3 Member Function Documentation

6.141.3.1 virtual void decaf::net::BufferedSocket::close () throw (decaf::io::IOException) [virtual]

Closes this object and deallocates the appropriate resources.

Exceptions

IOException

Implements decaf::io::Closeable (p. 951).

6.141.3.2 virtual void decaf::net::BufferedSocket::connect (const char * *host*, int *port*) throw (SocketException) [virtual]

Connects to the specified destination.

Closes this socket if connected to another destination.

Parameters

host The host of the server to connect to.

port The port of the server to connect to.

Exceptions

IOException Thrown if a failure occurred in the connect.

Implements decaf::net::Socket (p. 2787).

6.141.3.3 virtual io::InputStream* decaf::net::BufferedSocket::getInputStream () [inline, virtual]

Gets the InputStream for this socket.

Returns

The InputStream for this socket. NULL if not connected.

Implements decaf::net::Socket (p. 2788).

6.141.3.4 virtual bool decaf::net::BufferedSocket::getKeepAlive () const throw (SocketException) [inline, virtual]

Gets the keep alive flag.

Returns

True if keep alive is enabled.

Exceptions

SocketException (p. 2793) if the operation fails.

Implements **decaf::net::Socket** (p. 2788).

References **decaf::net::Socket::getKeepAlive()**.

6.141.3.5 `virtual io::OutputStream* decaf::net::BufferedSocket::getOutputStream () [inline, virtual]`

Gets the OutputStream for this socket.

Returns

the OutputStream for this socket. NULL if not connected.

Implements **decaf::net::Socket** (p. 2788).

6.141.3.6 `virtual int decaf::net::BufferedSocket::getReceiveBufferSize () const throw (SocketException) [inline, virtual]`

Gets the receive buffer size.

Returns

the receive buffer size in bytes.

Exceptions

SocketException (p. 2793) if the operation fails.

Implements **decaf::net::Socket** (p. 2788).

References **decaf::net::Socket::getReceiveBufferSize()**.

6.141.3.7 `virtual bool decaf::net::BufferedSocket::getReuseAddress () const throw (SocketException) [inline, virtual]`

Gets the reuse address flag.

Returns

True if the address can be reused.

Exceptions

SocketException (p. 2793) if the operation fails.

Implements **decaf::net::Socket** (p. 2789).

References **decaf::net::Socket::getReuseAddress()**.

6.141.3.8 `virtual int decaf::net::BufferedSocket::getSendBufferSize () const throw (SocketException) [inline, virtual]`

Gets the send buffer size.

Returns

the size in bytes of the send buffer.

Exceptions

SocketException (p. 2793) if the operation fails.

Implements **decaf::net::Socket** (p. 2789).

References `decaf::net::Socket::getSendBufferSize()`.

6.141.3.9 `virtual int decaf::net::BufferedSocket::getSoLinger () const throw (SocketException) [inline, virtual]`

Gets the linger time.

Returns

The linger time in seconds.

Exceptions

SocketException (p. 2793) if the operation fails.

Implements **decaf::net::Socket** (p. 2789).

References `decaf::net::Socket::getSoLinger()`.

6.141.3.10 `virtual int decaf::net::BufferedSocket::getSoTimeout () const throw (SocketException) [inline, virtual]`

Gets the timeout for socket operations.

Returns

The timeout in milliseconds for socket operations.

Exceptions

SocketException (p. 2793) Thrown if unable to retrieve the information.

Implements **decaf::net::Socket** (p. 2790).

References `decaf::net::Socket::getSoTimeout()`.

6.141.3.11 `virtual bool decaf::net::BufferedSocket::isConnected () const [inline, virtual]`

Indicates whether or not this socket is connected to a destination.

Returns

true if connected

Implements **decaf::net::Socket** (p. 2790).

References `decaf::net::Socket::isConnected()`.

6.141.3.12 `virtual void decaf::net::BufferedSocket::setKeepAlive (bool keepAlive) throw (SocketException) [inline, virtual]`

Enables/disables the keep alive flag.

Parameters

keepAlive If true, enables the flag.

Exceptions

SocketException (p. 2793) if the operation fails.

Implements **decaf::net::Socket** (p. 2790).

References `decaf::net::Socket::setKeepAlive()`.

6.141.3.13 `virtual void decaf::net::BufferedSocket::setReceiveBufferSize (int size) throw (SocketException) [inline, virtual]`

Sets the receive buffer size.

Parameters

size Number of bytes to set the receive buffer to.

Exceptions

SocketException (p. 2793) if the operation fails.

Implements **decaf::net::Socket** (p. 2790).

References `decaf::net::Socket::setReceiveBufferSize()`.

6.141.3.14 `virtual void decaf::net::BufferedSocket::setReuseAddress (bool reuse) throw (SocketException) [inline, virtual]`

Sets the reuse address flag.

Parameters

reuse If true, sets the flag.

Exceptions

SocketException (p. 2793) if the operation fails.

Implements **decaf::net::Socket** (p. 2791).

References `decaf::net::Socket::setReuseAddress()`.

6.141.3.15 `virtual void decaf::net::BufferedSocket::setSendBufferSize (int size)
throw (SocketException)` [inline, virtual]

Sets the send buffer size.

Parameters

size The number of bytes to set the send buffer to.

Exceptions

SocketException (p. 2793) if the operation fails.

Implements `decaf::net::Socket` (p. 2791).

References `decaf::net::Socket::setSendBufferSize()`.

6.141.3.16 `virtual void decaf::net::BufferedSocket::setSoLinger (int linger)
throw (SocketException)` [inline, virtual]

Sets the linger time.

Parameters

linger The linger time in seconds. If 0, linger is off.

Exceptions

SocketException (p. 2793) if the operation fails.

Implements `decaf::net::Socket` (p. 2791).

References `decaf::net::Socket::setSoLinger()`.

6.141.3.17 `virtual void decaf::net::BufferedSocket::setSoTimeout (int timeout)
throw (SocketException)` [inline, virtual]

Sets the timeout for socket operations.

Parameters

timeout The timeout in milliseconds for socket operations.

Exceptions

SocketException (p. 2793) Thrown if unable to set the information.

Implements `decaf::net::Socket` (p. 2792).

References `decaf::net::Socket::setSoTimeout()`.

The documentation for this class was generated from the following file:

- `src/main/decaf/net/BufferedSocket.h`

6.142 decaf::internal::nio::BufferFactory Class Reference

Factory class used by static methods in the **decaf::nio** (p.109) package to create the various default version of the NIO interfaces.

```
#include <src/main/decaf/internal/nio/BufferFactory.h>
```

Public Member Functions

- virtual **~BufferFactory** ()

Static Public Member Functions

- static **decaf::nio::ByteBuffer * createByteBuffer** (std::size_t capacity)
Allocates a new byte buffer whose position will be zero its limit will be its capacity and its mark is not set.
- static **decaf::nio::ByteBuffer * createByteBuffer** (unsigned char *buffer, std::size_t offset, std::size_t length) throw (lang::exceptions::NullPointerException)
Wraps the passed buffer with a new ByteBuffer.
- static **decaf::nio::ByteBuffer * createByteBuffer** (std::vector< unsigned char > &buffer)
Wraps the passed STL Byte Vector in a ByteBuffer.
- static **decaf::nio::CharBuffer * createCharBuffer** (std::size_t capacity)
Allocates a new char buffer whose position will be zero its limit will be its capacity and its mark is not set.
- static **decaf::nio::CharBuffer * createCharBuffer** (char *buffer, std::size_t offset, std::size_t length) throw (lang::exceptions::NullPointerException)
Wraps the passed buffer with a new CharBuffer.
- static **decaf::nio::CharBuffer * createCharBuffer** (std::vector< char > &buffer)
Wraps the passed STL Byte Vector in a CharBuffer.
- static **decaf::nio::DoubleBuffer * createDoubleBuffer** (std::size_t capacity)
Allocates a new double buffer whose position will be zero its limit will be its capacity and its mark is not set.
- static **decaf::nio::DoubleBuffer * createDoubleBuffer** (double *buffer, std::size_t offset, std::size_t length) throw (lang::exceptions::NullPointerException)
Wraps the passed buffer with a new DoubleBuffer.
- static **decaf::nio::DoubleBuffer * createDoubleBuffer** (std::vector< double > &buffer)
Wraps the passed STL Double Vector in a DoubleBuffer.
- static **decaf::nio::FloatBuffer * createFloatBuffer** (std::size_t capacity)
Allocates a new float buffer whose position will be zero its limit will be its capacity and its mark is not set.

- static **decaf::nio::FloatBuffer * createFloatBuffer** (float *buffer, std::size_t offset, std::size_t length) throw (lang::exceptions::NullPointerException)
Wraps the passed buffer with a new FloatBuffer.
- static **decaf::nio::FloatBuffer * createFloatBuffer** (std::vector< float > &buffer)
Wraps the passed STL Float Vector in a FloatBuffer.
- static **decaf::nio::LongBuffer * createLongBuffer** (std::size_t capacity)
Allocates a new long long buffer whose position will be zero its limit will be its capacity and its mark is not set.
- static **decaf::nio::LongBuffer * createLongBuffer** (long long *buffer, std::size_t offset, std::size_t length) throw (lang::exceptions::NullPointerException)
Wraps the passed buffer with a new LongBuffer.
- static **decaf::nio::LongBuffer * createLongBuffer** (std::vector< long long > &buffer)
Wraps the passed STL Long Vector in a LongBuffer.
- static **decaf::nio::IntBuffer * createIntBuffer** (std::size_t capacity)
Allocates a new int buffer whose position will be zero its limit will be its capacity and its mark is not set.
- static **decaf::nio::IntBuffer * createIntBuffer** (int *buffer, std::size_t offset, std::size_t length) throw (lang::exceptions::NullPointerException)
Wraps the passed buffer with a new IntBuffer.
- static **decaf::nio::IntBuffer * createIntBuffer** (std::vector< int > &buffer)
Wraps the passed STL int Vector in a IntBuffer.
- static **decaf::nio::ShortBuffer * createShortBuffer** (std::size_t capacity)
Allocates a new short buffer whose position will be zero its limit will be its capacity and its mark is not set.
- static **decaf::nio::ShortBuffer * createShortBuffer** (short *buffer, std::size_t offset, std::size_t length) throw (lang::exceptions::NullPointerException)
Wraps the passed buffer with a new ShortBuffer.
- static **decaf::nio::ShortBuffer * createShortBuffer** (std::vector< short > &buffer)
Wraps the passed STL Short Vector in a ShortBuffer.

6.142.1 Detailed Description

Factory class used by static methods in the **decaf::nio** (p.109) package to create the various default version of the NIO interfaces.

6.142.2 Constructor & Destructor Documentation

6.142.2.1 `virtual decaf::internal::nio::BufferFactory::~BufferFactory () [inline, virtual]`

6.142.3 Member Function Documentation

6.142.3.1 `static decaf::nio::ByteBuffer* decaf::internal::nio::BufferFactory::createByteBuffer (std::size_t capacity) [static]`

Allocates a new byte buffer whose position will be zero its limit will be its capacity and its mark is not set.

Parameters

capacity - the internal buffer's capacity.

Returns

a newly allocated ByteBuffer which the caller owns.

6.142.3.2 `static decaf::nio::ByteBuffer* decaf::internal::nio::BufferFactory::createByteBuffer (unsigned char * buffer, std::size_t offset, std::size_t length) throw (lang::exceptions::NullPointerException) [static]`

Wraps the passed buffer with a new ByteBuffer.

The new buffer will be backed by the given byte array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be array.length, its position will be offset, its limit will be offset + length, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

buffer - The array that will back the new buffer

offset - The offset of the subarray to be used

length - The length of the subarray to be used

Returns

a new ByteBuffer that is backed by buffer, caller owns.

6.142.3.3 `static decaf::nio::ByteBuffer* decaf::internal::nio::BufferFactory::createByteBuffer (std::vector< unsigned char > & buffer) [static]`

Wraps the passed STL Byte Vector in a ByteBuffer.

The new buffer will be backed by the given byte array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be buffer.size(), its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

buffer - The vector that will back the new buffer, the vector must have been sized to the desired size already by calling `vector.resize(N)`.

Returns

a new ByteBuffer that is backed by buffer, caller owns.

6.142.3.4 `static decaf::nio::CharBuffer* decaf::internal::nio::BufferFactory::createCharBuffer (std::size_t capacity) [static]`

Allocates a new char buffer whose position will be zero its limit will be its capacity and its mark is not set.

Parameters

capacity - the internal buffer's capacity.

Returns

a newly allocated CharBuffer which the caller owns.

6.142.3.5 `static decaf::nio::CharBuffer* decaf::internal::nio::BufferFactory::createCharBuffer (char * buffer, std::size_t offset, std::size_t length) throw (lang::exceptions::NullPointerException) [static]`

Wraps the passed buffer with a new CharBuffer.

The new buffer will be backed by the given byte array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be `array.length`, its position will be `offset`, its limit will be `offset + length`, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

buffer - The array that will back the new buffer

offset - The offset of the subarray to be used

length - The length of the subarray to be used

Returns

a new CharBuffer that is backed by buffer, caller owns.

6.142.3.6 `static decaf::nio::CharBuffer* decaf::internal::nio::BufferFactory::createCharBuffer (std::vector< char > & buffer) [static]`

Wraps the passed STL Byte Vector in a CharBuffer.

The new buffer will be backed by the given byte array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be `buffer.size()`, its

position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

buffer - The vector that will back the new buffer, the vector must have been sized to the desired size already by calling `vector.resize(N)`.

Returns

a new `CharBuffer` that is backed by *buffer*, caller owns.

6.142.3.7 `static decaf::nio::DoubleBuffer* decaf::internal::nio::BufferFactory::createDoubleBuffer (double * buffer, std::size_t offset, std::size_t length) throw (lang::exceptions::NullPointerException) [static]`

Wraps the passed buffer with a new `DoubleBuffer`.

The new buffer will be backed by the given byte array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be `array.length`, its position will be `offset`, its limit will be `offset + length`, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

buffer - The array that will back the new buffer
offset - The offset of the subarray to be used
length - The length of the subarray to be used

Returns

a new `DoubleBuffer` that is backed by *buffer*, caller owns.

6.142.3.8 `static decaf::nio::DoubleBuffer* decaf::internal::nio::BufferFactory::createDoubleBuffer (std::vector< double > & buffer) [static]`

Wraps the passed STL Double Vector in a `DoubleBuffer`.

The new buffer will be backed by the given byte array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be `buffer.size()`, its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

buffer - The vector that will back the new buffer, the vector must have been sized to the desired size already by calling `vector.resize(N)`.

Returns

a new `DoubleBuffer` that is backed by *buffer*, caller owns.

6.142.3.9 `static decaf::nio::DoubleBuffer* decaf::internal::nio::BufferFactory::createDoubleBuffer (std::size_t capacity) [static]`

Allocates a new double buffer whose position will be zero its limit will be its capacity and its mark is not set.

Parameters

capacity - the internal buffer's capacity.

Returns

a newly allocated DoubleBuffer which the caller owns.

6.142.3.10 `static decaf::nio::FloatBuffer* decaf::internal::nio::BufferFactory::createFloatBuffer (std::size_t capacity) [static]`

Allocates a new float buffer whose position will be zero its limit will be its capacity and its mark is not set.

Parameters

capacity - the internal buffer's capacity.

Returns

a newly allocated DoubleBuffer which the caller owns.

6.142.3.11 `static decaf::nio::FloatBuffer* decaf::internal::nio::BufferFactory::createFloatBuffer (float * buffer, std::size_t offset, std::size_t length) throw (lang::exceptions::NullPointerException) [static]`

Wraps the passed buffer with a new FloatBuffer.

The new buffer will be backed by the given byte array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be array.length, its position will be offset, its limit will be offset + length, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

buffer - The array that will back the new buffer

offset - The offset of the subarray to be used

length - The length of the subarray to be used

Returns

a new DoubleBuffer that is backed by buffer, caller owns.

6.142.3.12 `static decaf::nio::FloatBuffer* decaf::internal::nio::BufferFactory::createFloatBuffer (std::vector< float > & buffer) [static]`

Wraps the passed STL Float Vector in a FloatBuffer.

The new buffer will be backed by the given byte array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be `buffer.size()`, its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

buffer - The vector that will back the new buffer, the vector must have been sized to the desired size already by calling `vector.resize(N)`.

Returns

a new DoubleBuffer that is backed by *buffer*, caller owns.

6.142.3.13 `static decaf::nio::IntBuffer* decaf::internal::nio::BufferFactory::createIntBuffer (std::size_t capacity) [static]`

Allocates a new int buffer whose position will be zero its limit will be its capacity and its mark is not set.

Parameters

capacity - the internal buffer's capacity.

Returns

a newly allocated DoubleBuffer which the caller owns.

6.142.3.14 `static decaf::nio::IntBuffer* decaf::internal::nio::BufferFactory::createIntBuffer (std::vector< int > & buffer) [static]`

Wraps the passed STL int Vector in a IntBuffer.

The new buffer will be backed by the given byte array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be `buffer.size()`, its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

buffer - The vector that will back the new buffer, the vector must have been sized to the desired size already by calling `vector.resize(N)`.

Returns

a new DoubleBuffer that is backed by *buffer*, caller owns.

6.142.3.15 `static decaf::nio::IntBuffer* decaf::internal::nio::BufferFactory::createIntBuffer (int * buffer, std::size_t offset, std::size_t length) throw (lang::exceptions::NullPointerException) [static]`

Wraps the passed buffer with a new IntBuffer.

The new buffer will be backed by the given byte array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be array.length, its position will be offset, its limit will be offset + length, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

buffer - The array that will back the new buffer
offset - The offset of the subarray to be used
length - The length of the subarray to be used

Returns

a new DoubleBuffer that is backed by buffer, caller owns.

6.142.3.16 `static decaf::nio::LongBuffer* decaf::internal::nio::BufferFactory::createLongBuffer (std::vector< long long > & buffer) [static]`

Wraps the passed STL Long Vector in a LongBuffer.

The new buffer will be backed by the given byte array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be buffer.size(), its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

buffer - The vector that will back the new buffer, the vector must have been sized to the desired size already by calling vector.resize(N).

Returns

a new DoubleBuffer that is backed by buffer, caller owns.

6.142.3.17 `static decaf::nio::LongBuffer* decaf::internal::nio::BufferFactory::createLongBuffer (std::size_t capacity) [static]`

Allocates a new long long buffer whose position will be zero its limit will be its capacity and its mark is not set.

Parameters

capacity - the internal buffer's capacity.

Returns

a newly allocated DoubleBuffer which the caller owns.

6.142.3.18 `static decaf::nio::LongBuffer* decaf::internal::nio::BufferFactory::createLongBuffer (long long * buffer, std::size_t offset, std::size_t length) throw (lang::exceptions::NullPointerException) [static]`

Wraps the passed buffer with a new LongBuffer.

The new buffer will be backed by the given byte array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be array.length, its position will be offset, its limit will be offset + length, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

buffer - The array that will back the new buffer

offset - The offset of the subarray to be used

length - The length of the subarray to be used

Returns

a new DoubleBuffer that is backed by buffer, caller owns.

6.142.3.19 `static decaf::nio::ShortBuffer* decaf::internal::nio::BufferFactory::createShortBuffer (std::size_t capacity) [static]`

Allocates a new short buffer whose position will be zero its limit will be its capacity and its mark is not set.

Parameters

capacity - the internal buffer's capacity.

Returns

a newly allocated DoubleBuffer which the caller owns.

6.142.3.20 `static decaf::nio::ShortBuffer* decaf::internal::nio::BufferFactory::createShortBuffer (short * buffer, std::size_t offset, std::size_t length) throw (lang::exceptions::NullPointerException) [static]`

Wraps the passed buffer with a new ShortBuffer.

The new buffer will be backed by the given byte array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be array.length, its position will be offset, its limit will be offset + length, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

buffer - The array that will back the new buffer

offset - The offset of the subarray to be used

length - The length of the subarray to be used

Returns

a new DoubleBuffer that is backed by buffer, caller owns.

6.142.3.21 `static decaf::nio::ShortBuffer* decaf::internal::nio::BufferFactory::createShortBuffer (std::vector< short > & buffer) [static]`

Wraps the passed STL Short Vector in a ShortBuffer.

The new buffer will be backed by the given byte array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be buffer.size(), its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

buffer - The vector that will back the new buffer, the vector must have been sized to the desired size already by calling vector.resize(N).

Returns

a new DoubleBuffer that is backed by buffer, caller owns.

The documentation for this class was generated from the following file:

- src/main/decaf/internal/nio/**BufferFactory.h**

6.143 decaf::nio::BufferOverflowException Class Reference

```
#include <src/main/decaf/nio/BufferOverflowException.h>
```

Inheritance diagram for decaf::nio::BufferOverflowException:

Public Member Functions

- **BufferOverflowException** () throw ()
Default Constructor.
- **BufferOverflowException** (const lang::Exception &ex) throw ()
Copy Constructor.
- **BufferOverflowException** (const BufferOverflowException &ex) throw ()
Copy Constructor.
- **BufferOverflowException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()

Constructor - Initializes the file name and line number where this message occurred.

- **BufferOverflowException** (const std::exception **cause*) throw ()
Constructor.
- **BufferOverflowException** (const char **file*, const int *lineNumber*, const char **msg*,...) throw ()
Constructor.
- virtual **BufferOverflowException** * **clone** () const
Clones this exception.
- virtual ~**BufferOverflowException** () throw ()

6.143.1 Constructor & Destructor Documentation

6.143.1.1 decaf::nio::BufferOverflowException::BufferOverflowException ()
throw () [inline]

Default Constructor.

6.143.1.2 decaf::nio::BufferOverflowException::BufferOverflowException (const
lang::Exception & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex the exception to copy

6.143.1.3 decaf::nio::BufferOverflowException::BufferOverflowException (const
BufferOverflowException & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex the exception to copy, which is an instance of this type

6.143.1.4 decaf::nio::BufferOverflowException::BufferOverflowException (const
char * *file*, const int *lineNumber*, const std::exception * *cause*, const
char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.143.1.5 `decaf::nio::BufferOverflowException::BufferOverflowException (const std::exception * cause) throw () [inline]`

Constructor.

Parameters

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.143.1.6 `decaf::nio::BufferOverflowException::BufferOverflowException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor.

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.143.1.7 `virtual decaf::nio::BufferOverflowException::~~BufferOverflowException () throw () [inline, virtual]`

6.143.2 Member Function Documentation

6.143.2.1 `virtual BufferOverflowException* decaf::nio::BufferOverflowException::clone () const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Reimplemented from `decaf::lang::Exception` (p. 1479).

The documentation for this class was generated from the following file:

- `src/main/decaf/nio/BufferOverflowException.h`

6.144 decaf::nio::BufferUnderflowException Class Reference

```
#include <src/main/decaf/nio/BufferUnderflowException.h>
```

Inheritance diagram for decaf::nio::BufferUnderflowException:

Public Member Functions

- **BufferUnderflowException** () throw ()
Default Constructor.
- **BufferUnderflowException** (const lang::Exception &ex) throw ()
Copy Constructor.
- **BufferUnderflowException** (const **BufferUnderflowException** &ex) throw ()
Copy Constructor.
- **BufferUnderflowException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **BufferUnderflowException** (const std::exception *cause) throw ()
Constructor.
- **BufferUnderflowException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor.
- virtual **BufferUnderflowException** * clone () const
Clones this exception.
- virtual ~**BufferUnderflowException** () throw ()

6.144.1 Constructor & Destructor Documentation

6.144.1.1 decaf::nio::BufferUnderflowException::BufferUnderflowException () throw () [inline]

Default Constructor.

6.144.1.2 decaf::nio::BufferUnderflowException::BufferUnderflowException (const lang::Exception & ex) throw () [inline]

Copy Constructor.

Parameters

ex the exception to copy

6.144.1.3 `decaf::nio::BufferUnderflowException::BufferUnderflowException (const BufferUnderflowException & ex) throw () [inline]`

Copy Constructor.

Parameters

ex the exception to copy, which is an instance of this type

6.144.1.4 `decaf::nio::BufferUnderflowException::BufferUnderflowException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.144.1.5 `decaf::nio::BufferUnderflowException::BufferUnderflowException (const std::exception * cause) throw () [inline]`

Constructor.

Parameters

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.144.1.6 `decaf::nio::BufferUnderflowException::BufferUnderflowException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor.

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.144.1.7 virtual
decaf::nio::BufferUnderflowException::~~BufferUnderflowException ()
throw () [inline, virtual]

6.144.2 Member Function Documentation

6.144.2.1 virtual BufferUnderflowException* decaf::nio::BufferUnderflowException::clone () const
 [inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Reimplemented from **decaf::lang::Exception** (p. 1479).

The documentation for this class was generated from the following file:

- src/main/decaf/nio/**BufferUnderflowException.h**

6.145 decaf::lang::Byte Class Reference

```
#include <src/main/decaf/lang/Byte.h>
```

Inheritance diagram for decaf::lang::Byte:

Public Member Functions

- **Byte** (unsigned char value)
- **Byte** (const std::string &value) throw (exceptions::NumberFormatException)
- virtual **~Byte** ()
- virtual int **compareTo** (const **Byte** &c) const
*Compares this **Byte** (p. 776) instance with another.*
- virtual bool **operator==** (const **Byte** &c) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const **Byte** &c) const
Compares this object to another and returns true if this object is considered to be less than the one passed.
- virtual int **compareTo** (const unsigned char &c) const
*Compares this **Byte** (p. 776) instance with a char type.*
- virtual bool **operator==** (const unsigned char &c) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const unsigned char &c) const

Compares this object to another and returns true if this object is considered to be less than the one passed.

- bool **equals** (const **Byte** &c) const
- bool **equals** (const unsigned char &c) const
- std::string **toString** () const
- virtual double **doubleValue** () const

Answers the double value which the receiver represents.

- virtual float **floatValue** () const

Answers the float value which the receiver represents.

- virtual unsigned char **byteValue** () const

Answers the byte value which the receiver represents.

- virtual short **shortValue** () const

Answers the short value which the receiver represents.

- virtual int **intValue** () const

Answers the int value which the receiver represents.

- virtual long long **longValue** () const

Answers the long value which the receiver represents.

Static Public Member Functions

- static std::string **toString** (unsigned char value)
- static **Byte decode** (const std::string &value) throw (exceptions::NumberFormatException)

*Decodes a String into a **Byte** (p. 776).*

- static unsigned char **parseByte** (const std::string &s, int radix) throw (exceptions::NumberFormatException)

Parses the string argument as a signed unsigned char in the radix specified by the second argument.

- static unsigned char **parseByte** (const std::string &s) throw (exceptions::NumberFormatException)

Parses the string argument as a signed decimal unsigned char.

- static **Byte valueOf** (unsigned char value)

*Returns a **Character** (p. 914) instance representing the specified char value.*

- static **Byte valueOf** (const std::string &value) throw (exceptions::NumberFormatException)

*Returns a **Byte** (p. 776) object holding the value given by the specified std::string.*

- static **Byte valueOf** (const std::string &value, int radix) throw (exceptions::NumberFormatException)

*Returns a **Byte** (p. 776) object holding the value extracted from the specified `std::string` when parsed with the radix given by the second argument.*

Static Public Attributes

- static const unsigned char **MIN_VALUE** = 0x7F
The minimum value that a unsigned char can take on.
- static const unsigned char **MAX_VALUE** = 0x80
The maximum value that a unsigned char can take on.
- static const int **SIZE** = 8
The size of the primitive charactor in bits.

6.145.1 Constructor & Destructor Documentation

6.145.1.1 decaf::lang::Byte::Byte (unsigned char value)

Parameters

value - the primitive value to wrap

6.145.1.2 decaf::lang::Byte::Byte (const std::string & value) throw (exceptions::NumberFormatException)

Parameters

value - the string to convert to an unsigned char

Exceptions

NumberFormatException

6.145.1.3 virtual decaf::lang::Byte::~~Byte () [inline, virtual]

6.145.2 Member Function Documentation

6.145.2.1 virtual unsigned char decaf::lang::Byte::byteValue () const [inline, virtual]

Answers the byte value which the receiver represents.

Returns

byte the value of the receiver.

6.145.2.2 `virtual int decaf::lang::Byte::compareTo (const unsigned char & c)`
`const [inline, virtual]`

Compares this **Byte** (p. 776) instance with a char type.

Parameters

c - the char instance to be compared

Returns

zero if this object represents the same char value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

Implements **decaf::lang::Comparable**< **unsigned char** > (p. 1010).

6.145.2.3 `virtual int decaf::lang::Byte::compareTo (const Byte & c) const`
`[inline, virtual]`

Compares this **Byte** (p. 776) instance with another.

Parameters

c - the **Byte** (p. 776) instance to be compared

Returns

zero if this object represents the same char value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

6.145.2.4 `static Byte decaf::lang::Byte::decode (const std::string & value) throw`
`(exceptions::NumberFormatException) [static]`

Decodes a String into a **Byte** (p. 776).

Accepts decimal, hexadecimal, and octal numbers given by the following grammar:

The sequence of characters following an (optional) negative sign and/or radix specifier ("0x", "0X", "#", or leading zero) is parsed as by the **Byte::parseByte** (p. 782) method with the indicated radix (10, 16, or 8). This sequence of characters must represent a positive value or a **NumberFormatException** will be thrown. The result is negated if first character of the specified String is the minus sign. No whitespace characters are permitted in the string.

Parameters

value - The string to decode

Returns

a **Byte** (p. 776) object containing the decoded value

Exceptions

NumberFormatException if the string is not formatted correctly.

6.145.2.5 `virtual double decaf::lang::Byte::doubleValue () const [inline, virtual]`

Answers the double value which the receiver represents.

Returns

double the value of the receiver.

6.145.2.6 `bool decaf::lang::Byte::equals (const unsigned char & c) const [inline, virtual]`

Returns

true if the two Bytes have the same value.

Implements `decaf::lang::Comparable< unsigned char >` (p. 1010).

6.145.2.7 `bool decaf::lang::Byte::equals (const Byte & c) const [inline]`

Returns

true if the two **Byte** (p. 776) Objects have the same value.

6.145.2.8 `virtual float decaf::lang::Byte::floatValue () const [inline, virtual]`

Answers the float value which the receiver represents.

Returns

float the value of the receiver.

6.145.2.9 `virtual int decaf::lang::Byte::intValue () const [inline, virtual]`

Answers the int value which the receiver represents.

Returns

int the value of the receiver.

6.145.2.10 `virtual long long decaf::lang::Byte::longValue () const [inline, virtual]`

Answers the long value which the receiver represents.

Returns

long long the value of the receiver.

6.145.2.11 `virtual bool decaf::lang::Byte::operator< (const unsigned char & c) const [inline, virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

Parameters

`c` - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

Implements `decaf::lang::Comparable< unsigned char >` (p. 1011).

6.145.2.12 `virtual bool decaf::lang::Byte::operator< (const Byte & c) const [inline, virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

Parameters

`c` - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

6.145.2.13 `virtual bool decaf::lang::Byte::operator== (const Byte & c) const [inline, virtual]`

Compares equality between this object and the one passed.

Parameters

`c` - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

6.145.2.14 `virtual bool decaf::lang::Byte::operator== (const unsigned char & c) const [inline, virtual]`

Compares equality between this object and the one passed.

Parameters

`c` - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable**< unsigned char > (p.1011).

6.145.2.15 static unsigned char decaf::lang::Byte::parseByte (const std::string & s) throw (exceptions::NumberFormatException) [static]

Parses the string argument as a signed decimal unsigned char.

The characters in the string must all be decimal digits, except that the first character may be an ASCII minus sign '-' to indicate a negative value. The resulting unsigned char value is returned, exactly as if the argument and the radix 10 were given as arguments to the parseByte(const std::string, int) method.

Parameters

s - String to convert to a unsigned char

Returns

the converted unsigned char value

Exceptions

NumberFormatException if the string is not a unsigned char.

6.145.2.16 static unsigned char decaf::lang::Byte::parseByte (const std::string & s, int radix) throw (exceptions::NumberFormatException) [static]

Parses the string argument as a signed unsigned char in the radix specified by the second argument.

The characters in the string must all be digits, of the specified radix (as determined by whether **Character.digit(char, int)** (p.917) returns a nonnegative value) except that the first character may be an ASCII minus sign '-' to indicate a negative value. The resulting byte value is returned.

An exception of type *NumberFormatException* is thrown if any of the following situations occurs:
* The first argument is null or is a string of length zero. * The radix is either smaller than **Character.MIN_RADIX** (p.921) or larger than **Character::MAX_RADIX** (p.921). * Any character of the string is not a digit of the specified radix, except that the first character may be a minus sign '-' provided that the string is longer than length 1. * The value represented by the string is not a value of type unsigned char.

Parameters

s - the String containing the unsigned char to be parsed

radix - the radix to be used while parsing s

Returns

the unsigned char represented by the string argument in the specified radix.

Exceptions

NumberFormatException - If String does not contain a parsable unsigned char.

6.145.2.17 `virtual short decaf::lang::Byte::shortValue () const` [inline, virtual]

Answers the short value which the receiver represents.

Returns

short the value of the receiver.

6.145.2.18 `static std::string decaf::lang::Byte::toString (unsigned char value)`
[static]**Returns**

a string representing the primitive value as Base 10

6.145.2.19 `std::string decaf::lang::Byte::toString () const`**Returns**

this **Byte** (p. 776) Object as a String Representation

6.145.2.20 `static Byte decaf::lang::Byte::valueOf (unsigned char value)`
[inline, static]

Returns a **Character** (p. 914) instance representing the specified char value.

Parameters

value - the primitive char to wrap.

Returns

a new **Character** (p. 914) instance that wraps this value.

6.145.2.21 `static Byte decaf::lang::Byte::valueOf (const std::string & value, int radix) throw (exceptions::NumberFormatException)` [static]

Returns a **Byte** (p. 776) object holding the value extracted from the specified std::string when parsed with the radix given by the second argument.

The first argument is interpreted as representing a signed unsigned char in the radix specified by the second argument, exactly as if the argument were given to the `parseByte(std::string, int)` method. The result is a **Byte** (p. 776) object that represents the unsigned char value specified by the string.

Parameters

value - std::string to parse as base (radix)

radix - base of the string to parse.

Returns

new **Byte** (p. 776) Object wrapping the primitive

Exceptions

NumberFormatException if the string is not a valid unsigned char.

6.145.2.22 `static Byte decaf::lang::Byte::valueOf (const std::string & value)
throw (exceptions::NumberFormatException) [static]`

Returns a **Byte** (p. 776) object holding the value given by the specified std::string.

The argument is interpreted as representing a signed decimal unsigned char, exactly as if the argument were given to the `parseByte(std::string)` method. The result is a **Byte** (p. 776) object that represents the unsigned char value specified by the string.

Parameters

value - std::string to parse as base 10

Returns

new **Byte** (p. 776) Object wrapping the primitive

Exceptions

NumberFormatException if the string is not a decimal unsigned char.

6.145.3 Field Documentation

6.145.3.1 `const unsigned char decaf::lang::Byte::MAX_VALUE = 0x80 [static]`

The maximum value that a unsigned char can take on.

6.145.3.2 `const unsigned char decaf::lang::Byte::MIN_VALUE = 0x7F [static]`

The minimum value that a unsigned char can take on.

6.145.3.3 `const int decaf::lang::Byte::SIZE = 8 [static]`

The size of the primitive charactor in bits.

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Byte.h`

6.146 decaf::internal::util::ByteArrayAdapter Class Reference

This class adapts primitive type arrays to a base byte array so that the classes can inter-operate on the same base byte array without copying data.

```
#include <src/main/decaf/internal/util/ByteArrayAdapter.h>
```

Inheritance diagram for `decaf::internal::util::ByteArrayAdapter`:

Data Structures

- union **Array**

Public Member Functions

- **ByteArrayAdapter** (std::size_t capacity)
Creates a byte array object that is allocated internally and is then owned and deleted when this object is deleted.
- **ByteArrayAdapter** (unsigned char *array, std::size_t capacity, bool own=false) throw (lang::exceptions::NullPointerException)
Creates a byte array object that wraps the given array.
- **ByteArrayAdapter** (char *array, std::size_t capacity, bool own=false) throw (lang::exceptions::NullPointerException)
Creates a byte array object that wraps the given array.
- **ByteArrayAdapter** (double *array, std::size_t capacity, bool own=false) throw (lang::exceptions::NullPointerException)
Creates a byte array object that wraps the given array.
- **ByteArrayAdapter** (float *array, std::size_t capacity, bool own=false) throw (lang::exceptions::NullPointerException)
Creates a byte array object that wraps the given array.
- **ByteArrayAdapter** (long long *array, std::size_t capacity, bool own=false) throw (lang::exceptions::NullPointerException)
Creates a byte array object that wraps the given array.
- **ByteArrayAdapter** (int *array, std::size_t capacity, bool own=false) throw (lang::exceptions::NullPointerException)
Creates a byte array object that wraps the given array.
- **ByteArrayAdapter** (short *array, std::size_t capacity, bool own=false) throw (lang::exceptions::NullPointerException)
Creates a byte array object that wraps the given array.
- virtual ~**ByteArrayAdapter** ()
- virtual std::size_t **getCapacity** () const
Gets the capacity of the underlying array.
- virtual std::size_t **getCharCapacity** () const
Gets the capacity of the underlying array as if it contains chars.
- virtual std::size_t **getDoubleCapacity** () const
Gets the capacity of the underlying array as if it contains doubles.
- virtual std::size_t **getFloatCapacity** () const
Gets the capacity of the underlying array as if it contains doubles.

- virtual std::size_t **getLongCapacity** () const
Gets the capacity of the underlying array as if it contains doubles.
- virtual std::size_t **getIntCapacity** () const
Gets the capacity of the underlying array as if it contains ints.
- virtual std::size_t **getShortCapacity** () const
Gets the capacity of the underlying array as if it contains shorts.
- virtual unsigned char * **getByteArray** ()
Gets the pointer to the array we are wrapping.
- virtual char * **getCharArray** ()
Gets the pointer to the array we are wrapping.
- virtual short * **getShortArray** ()
Gets the pointer to the array we are wrapping.
- virtual int * **getIntArray** ()
Gets the pointer to the array we are wrapping.
- virtual long long * **getLongArray** ()
Gets the pointer to the array we are wrapping.
- virtual double * **getDoubleArray** ()
Gets the pointer to the array we are wrapping.
- virtual float * **getFloatArray** ()
Gets the pointer to the array we are wrapping.
- virtual void **read** (unsigned char *buffer, std::size_t offset, std::size_t length) const throw (decaf::lang::exceptions::NullPointerException, decaf::nio::BufferUnderflowException)
Reads from the Byte array starting at the specified offset and reading the specified length.
- virtual void **write** (unsigned char *buffer, std::size_t offset, std::size_t length) throw (decaf::lang::exceptions::NullPointerException, decaf::nio::BufferOverflowException)
Writes from the Byte array given, starting at the specified offset and writing the specified amount of data into this objects internal array.
- virtual void **resize** (std::size_t capacity) throw (lang::exceptions::InvalidStateException)
Resizes the underlying array to the new given capacity, preserving all the Data that was previously in the array, unless the resize is smaller than the current size in which case only the data that will fit into the new array is preserved.
- virtual void **clear** ()
Clear all data from that Array, setting the underlying bytes to zero.
- unsigned char & **operator[]** (std::size_t index) throw (decaf::lang::exceptions::IndexOutOfBoundsException)

*Allows the **ByteArrayAdapter** (p. 784) to be indexed as a standard array.*

- `const unsigned char & operator[] (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException)`
- `virtual unsigned char get (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)`

Absolute get method.

- `virtual char getChar (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)`

Reads one byte at the given index and returns it.

- `virtual double getDouble (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)`

Reads eight bytes at the given index and returns it.

- `virtual double getDoubleAt (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)`

Reads eight bytes at the given byte index and returns it.

- `virtual float getFloat (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)`

Reads four bytes at the given index and returns it.

- `virtual float getFloatAt (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)`

Reads four bytes at the given byte index and returns it.

- `virtual long long getLong (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)`

Reads eight bytes at the given index and returns it.

- `virtual long long getLongAt (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)`

Reads eight bytes at the given byte index and returns it.

- `virtual int getInt (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)`

Reads four bytes at the given index and returns it.

- `virtual int getIntAt (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)`

Reads four bytes at the given byte index and returns it.

- `virtual short getShort (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)`

Reads two bytes at the given index and returns it.

- `virtual short getShortAt (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)`

Reads two bytes at the given byte index and returns it.

- virtual **ByteArrayAdapter** & **put** (std::size_t index, unsigned char value) throw (lang::exceptions::IndexOutOfBoundsException)
Writes the given byte into this buffer at the given index.
- virtual **ByteArrayAdapter** & **putChar** (std::size_t index, char value) throw (lang::exceptions::IndexOutOfBoundsException)
Writes one byte containing the given value, into this buffer at the given index.
- virtual **ByteArrayAdapter** & **putDouble** (std::size_t index, double value) throw (lang::exceptions::IndexOutOfBoundsException)
Writes eight bytes containing the given value, into this buffer at the given index.
- virtual **ByteArrayAdapter** & **putDoubleAt** (std::size_t index, double value) throw (lang::exceptions::IndexOutOfBoundsException)
Writes eight bytes containing the given value, into this buffer at the given byte index.
- virtual **ByteArrayAdapter** & **putFloat** (std::size_t index, float value) throw (lang::exceptions::IndexOutOfBoundsException)
Writes four bytes containing the given value, into this buffer at the given index.
- virtual **ByteArrayAdapter** & **putFloatAt** (std::size_t index, float value) throw (lang::exceptions::IndexOutOfBoundsException)
Writes four bytes containing the given value, into this buffer at the given byte index.
- virtual **ByteArrayAdapter** & **putLong** (std::size_t index, long long value) throw (lang::exceptions::IndexOutOfBoundsException)
Writes eight bytes containing the given value, into this buffer at the given index.
- virtual **ByteArrayAdapter** & **putLongAt** (std::size_t index, long long value) throw (lang::exceptions::IndexOutOfBoundsException)
Writes eight bytes containing the given value, into this buffer at the given byte index.
- virtual **ByteArrayAdapter** & **putInt** (std::size_t index, int value) throw (lang::exceptions::IndexOutOfBoundsException)
Writes four bytes containing the given value, into this buffer at the given index.
- virtual **ByteArrayAdapter** & **putIntAt** (std::size_t index, int value) throw (lang::exceptions::IndexOutOfBoundsException)
Writes four bytes containing the given value, into this buffer at the given byte index.
- virtual **ByteArrayAdapter** & **putShort** (std::size_t index, short value) throw (lang::exceptions::IndexOutOfBoundsException)
Writes two bytes containing the given value, into this buffer at the given index.
- virtual **ByteArrayAdapter** & **putShortAt** (std::size_t index, short value) throw (lang::exceptions::IndexOutOfBoundsException)
Writes two bytes containing the given value, into this buffer at the given byte index.

6.146.1 Detailed Description

This class adapts primitive type arrays to a base byte array so that the classes can inter-operate on the same base byte array without copying data. All the array types are mapped down to a byte array and methods are supplied for accessing the data in any of the primitive type forms.

Methods in this class that do not return a specific value return a reference to this object so that calls can be chained.

6.146.2 Constructor & Destructor Documentation

6.146.2.1 `decaf::internal::util::ByteArrayAdapter::ByteArrayAdapter (std::size_t capacity)`

Creates a byte array object that is allocated internally and is then owned and deleted when this object is deleted.

The array is initially created with all elements initialized to zero.

Parameters

capacity - size of the array, this is the limit we read and write to.

6.146.2.2 `decaf::internal::util::ByteArrayAdapter::ByteArrayAdapter (unsigned char * array, std::size_t capacity, bool own = false) throw (lang::exceptions::NullPointerException)`

Creates a byte array object that wraps the given array.

If the own flag is set then it will delete this array when this object is deleted.

Parameters

array - array to wrap

capacity - size of the array, this is the limit we read and write to.

own - is this class now the owner of the pointer.

Exceptions

NullPointerException if buffer is NULL

6.146.2.3 `decaf::internal::util::ByteArrayAdapter::ByteArrayAdapter (char * array, std::size_t capacity, bool own = false) throw (lang::exceptions::NullPointerException)`

Creates a byte array object that wraps the given array.

If the own flag is set then it will delete this array when this object is deleted.

Parameters

array - array to wrap

capacity - size of the array, this is the limit we read and write to.

own - is this class now the owner of the pointer.

Exceptions

NullPointerException if buffer is NULL

6.146.2.4 `decaf::internal::util::ByteArrayAdapter::ByteArrayAdapter (double * array, std::size_t capacity, bool own = false) throw (lang::exceptions::NullPointerException)`

Creates a byte array object that wraps the given array.

If the own flag is set then it will delete this array when this object is deleted.

Parameters

array - array to wrap

capacity - size of the array, this is the limit we read and write to.

own - is this class now the owner of the pointer.

Exceptions

NullPointerException if buffer is NULL

6.146.2.5 `decaf::internal::util::ByteArrayAdapter::ByteArrayAdapter (float * array, std::size_t capacity, bool own = false) throw (lang::exceptions::NullPointerException)`

Creates a byte array object that wraps the given array.

If the own flag is set then it will delete this array when this object is deleted.

Parameters

array - array to wrap

capacity - size of the array, this is the limit we read and write to.

own - is this class now the owner of the pointer.

Exceptions

NullPointerException if buffer is NULL

6.146.2.6 `decaf::internal::util::ByteArrayAdapter::ByteArrayAdapter (long long * array, std::size_t capacity, bool own = false) throw (lang::exceptions::NullPointerException)`

Creates a byte array object that wraps the given array.

If the own flag is set then it will delete this array when this object is deleted.

Parameters

array - array to wrap

capacity - size of the array, this is the limit we read and write to.

own - is this class now the owner of the pointer.

Exceptions

NullPointerException if buffer is NULL

6.146.2.7 `decaf::internal::util::ByteArrayAdapter::ByteArrayAdapter (int
* array, std::size_t capacity, bool own = false) throw (
lang::exceptions::NullPointerException)`

Creates a byte array object that wraps the given array.

If the own flag is set then it will delete this array when this object is deleted.

Parameters

array - array to wrap

capacity - size of the array, this is the limit we read and write to.

own - is this class now the owner of the pointer.

Exceptions

NullPointerException if buffer is NULL

6.146.2.8 `decaf::internal::util::ByteArrayAdapter::ByteArrayAdapter (short
* array, std::size_t capacity, bool own = false) throw (
lang::exceptions::NullPointerException)`

Creates a byte array object that wraps the given array.

If the own flag is set then it will delete this array when this object is deleted.

Parameters

array - array to wrap

capacity - size of the array, this is the limit we read and write to.

own - is this class now the owner of the pointer.

Exceptions

NullPointerException if buffer is NULL

6.146.2.9 `virtual decaf::internal::util::ByteArrayAdapter::~~ByteArrayAdapter ()
[virtual]`

6.146.3 Member Function Documentation

6.146.3.1 `virtual void decaf::internal::util::ByteArrayAdapter::clear () [virtual]`

Clear all data from that Array, setting the underlying bytes to zero.

6.146.3.2 `virtual unsigned char decaf::internal::util::ByteArrayAdapter::get (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)`
[virtual]

Absolute get method.

Reads the byte at the given index.

Parameters

index - the index in the Buffer where the byte is to be read

Returns

the byte that is located at the given index

Exceptions

IndexOutOfBoundsException - If index is not smaller than the buffer's limit

6.146.3.3 `virtual unsigned char* decaf::internal::util::ByteArrayAdapter::getByteArray ()`
[inline, virtual]

Gets the pointer to the array we are wrapping.

Changes to the data in this array are reflected by all **ByteArrayAdapter** (p. 784) objects that point to this array.

Returns

an unsigned char* pointer to the array this object wraps.

6.146.3.4 `virtual std::size_t decaf::internal::util::ByteArrayAdapter::getCapacity () const` [inline, virtual]

Gets the capacity of the underlying array.

Returns

the size the array.

6.146.3.5 `virtual char decaf::internal::util::ByteArrayAdapter::getChar (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)`
[inline, virtual]

Reads one byte at the given index and returns it.

Parameters

index - the index in the Buffer where the byte is to be read

Returns

the char at the given index in the buffer

Exceptions

IndexOutOfBoundsException - If index is not smaller than the buffer's limit

6.146.3.6 `virtual char* decaf::internal::util::ByteArrayAdapter::getCharArray ()`
[inline, virtual]

Gets the pointer to the array we are wrapping.

Changes to the data in this array are reflected by all **ByteArrayAdapter** (p. 784) objects that point to this array.

Returns

an char* pointer to the array this object wraps.

6.146.3.7 `virtual std::size_t decaf::internal::util::ByteArrayAdapter::getCharCapacity () const` [inline, virtual]

Gets the capacity of the underlying array as if it contains chars.

Returns

the size the array.

6.146.3.8 `virtual double decaf::internal::util::ByteArrayAdapter::getDouble (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)` [virtual]

Reads eight bytes at the given index and returns it.

The index is a relative to the size of the type to be read, in otherwords when accessing the element in the array `index * sizeof(type)` if the actual start index of the type to be read.

Parameters

index - the index in the Buffer where the bytes are to be read

Returns

the double at the given index in the buffer

Exceptions

IndexOutOfBoundsException - If there are not enough bytes remaining to fill the requested Data Type

**6.146.3.9 virtual double* decaf::internal::util::ByteArrayAdapter::getDoubleArray
() [inline, virtual]**

Gets the pointer to the array we are wrapping.

Changes to the data in this array are reflected by all **ByteArrayAdapter** (p. 784) objects that point to this array.

Returns

an double* pointer to the array this object wraps.

**6.146.3.10 virtual double decaf::internal::util::ByteArrayAdapter::getDoubleAt
(std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)
[virtual]**

Reads eight bytes at the given byte index and returns it.

Parameters

index - the index in the Buffer where the bytes are to be read

Returns

the double at the given index in the buffer

Exceptions

IndexOutOfBoundsException - If there are not enough bytes remaining to fill the requested Data Type

**6.146.3.11 virtual std::size_t decaf::internal::util::ByteArrayAdapter::getDoubleCapacity
() const [inline, virtual]**

Gets the capacity of the underlying array as if it contains doubles.

Returns

the size the array.

**6.146.3.12 virtual float decaf::internal::util::ByteArrayAdapter::getFloat
(std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)
[virtual]**

Reads four bytes at the given index and returns it.

The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof(type)` if the actual start index of the type to be read.

Parameters

index - the index in the Buffer where the bytes are to be read

Returns

the float at the given index in the buffer

Exceptions

IndexOutOfBoundsException - If there are not enough bytes remaining to fill the requested Data Type

6.146.3.13 `virtual float* decaf::internal::util::ByteArrayAdapter::getFloatArray () [inline, virtual]`

Gets the pointer to the array we are wrapping.

Changes to the data in this array are reflected by all **ByteArrayAdapter** (p.784) objects that point to this array.

Returns

an float* pointer to the array this object wraps.

6.146.3.14 `virtual float decaf::internal::util::ByteArrayAdapter::getFloatAt (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException) [virtual]`

Reads four bytes at the given byte index and returns it.

Parameters

index - the index in the Buffer where the bytes are to be read

Returns

the float at the given index in the buffer

Exceptions

IndexOutOfBoundsException - If there are not enough bytes remaining to fill the requested Data Type

6.146.3.15 `virtual std::size_t decaf::internal::util::ByteArrayAdapter::getFloatCapacity () const [inline, virtual]`

Gets the capacity of the underlying array as if it contains doubles.

Returns

the size the array.

6.146.3.16 `virtual int decaf::internal::util::ByteArrayAdapter::getInt (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)`
[virtual]

Reads four bytes at the given index and returns it.

The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof(type)` if the actual start index of the type to be read.

Parameters

index - the index in the Buffer where the bytes are to be read

Returns

the int at the given index in the buffer

Exceptions

IndexOutOfBoundsException - If there are not enough bytes remaining to fill the requested Data Type

6.146.3.17 `virtual int* decaf::internal::util::ByteArrayAdapter::getIntArray ()`
[inline, virtual]

Gets the pointer to the array we are wrapping.

Changes to the data in this array are reflected by all **ByteArrayAdapter** (p. 784) objects that point to this array.

Returns

an int* pointer to the array this object wraps.

6.146.3.18 `virtual int decaf::internal::util::ByteArrayAdapter::getIntAt (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)`
[virtual]

Reads four bytes at the given byte index and returns it.

Parameters

index - the index in the Buffer where the bytes are to be read

Returns

the int at the given index in the buffer

Exceptions

IndexOutOfBoundsException - If there are not enough bytes remaining to fill the requested Data Type

6.146.3.19 `virtual std::size_t decaf::internal::util::ByteArrayAdapter::getIntCapacity () const [inline, virtual]`

Gets the capacity of the underlying array as if it contains ints.

Returns

the size the array.

6.146.3.20 `virtual long long decaf::internal::util::ByteArrayAdapter::getLong (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException) [virtual]`

Reads eight bytes at the given index and returns it.

The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof(type)` if the actual start index of the type to be read.

Parameters

index - the index in the Buffer where the bytes are to be read

Returns

the long long at the given index in the buffer

Exceptions

IndexOutOfBoundsException - If there are not enough bytes remaining to fill the requested Data Type

6.146.3.21 `virtual long long* decaf::internal::util::ByteArrayAdapter::getLongArray () [inline, virtual]`

Gets the pointer to the array we are wrapping.

Changes to the data in this array are reflected by all **ByteArrayAdapter** (p. 784) objects that point to this array.

Returns

an long long* pointer to the array this object wraps.

6.146.3.22 `virtual long long decaf::internal::util::ByteArrayAdapter::getLongAt (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException) [virtual]`

Reads eight bytes at the given byte index and returns it.

Parameters

index - the index in the Buffer where the bytes are to be read

Returns

the long long at the given index in the buffer

Exceptions

IndexOutOfBoundsException - If there are not enough bytes remaining to fill the requested Data Type

6.146.3.23 `virtual std::size_t decaf::internal::util::ByteArrayAdapter::getLongCapacity () const [inline, virtual]`

Gets the capacity of the underlying array as if it contains doubles.

Returns

the size the array.

6.146.3.24 `virtual short decaf::internal::util::ByteArrayAdapter::getShort (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException) [virtual]`

Reads two bytes at the given index and returns it.

The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof(type)` if the actual start index of the type to be read.

Parameters

index - the index in the Buffer where the bytes are to be read

Returns

the short at the given index in the buffer

Exceptions

IndexOutOfBoundsException - If there are not enough bytes remaining to fill the requested Data Type

6.146.3.25 `virtual short* decaf::internal::util::ByteArrayAdapter::getShortArray () [inline, virtual]`

Gets the pointer to the array we are wrapping.

Changes to the data in this array are reflected by all **ByteArrayAdapter** (p. 784) objects that point to this array.

Returns

an short* pointer to the array this object wraps.

6.146.3.26 `virtual short decaf::internal::util::ByteArrayAdapter::getShortAt (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException) [virtual]`

Reads two bytes at the given byte index and returns it.

Parameters

index - the index in the Buffer where the bytes are to be read

Returns

the short at the given index in the buffer

Exceptions

IndexOutOfBoundsException - If there are not enough bytes remaining to fill the requested Data Type

6.146.3.27 `virtual std::size_t decaf::internal::util::ByteArrayAdapter::getShortCapacity () const [inline, virtual]`

Gets the capacity of the underlying array as if it contains shorts.

Returns

the size the array.

6.146.3.28 `unsigned char& decaf::internal::util::ByteArrayAdapter::operator[] (std::size_t index) throw (decaf::lang::exceptions::IndexOutOfBoundsException)`

Allows the **ByteArrayAdapter** (p. 784) to be indexed as a standard array. calling the non const version allows the user to change the value at index

Parameters

index - the position in the array to access

Exceptions

IndexOutOfBoundsException

6.146.3.29 `const unsigned char& decaf::internal::util::ByteArrayAdapter::operator[]
 (std::size_t index) const throw (de-
 caf::lang::exceptions::IndexOutOfBoundsException
)`

6.146.3.30 `virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::put
 (std::size_t index, unsigned char value) throw (
 lang::exceptions::IndexOutOfBoundsException) [virtual]`

Writes the given byte into this buffer at the given index.

The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof(type)` if the actual start index of the type to be read.

Parameters

index - position in the Buffer to write the data

value - the byte to write.

Returns

a reference to this buffer

Exceptions

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

6.146.3.31 `virtual ByteArrayAdapter& de-
 caf::internal::util::ByteArrayAdapter::putChar (std::size_t index,
 char value) throw (lang::exceptions::IndexOutOfBoundsException)
 [virtual]`

Writes one byte containing the given value, into this buffer at the given index.

The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof(type)` if the actual start index of the type to be read.

Parameters

index - position in the Buffer to write the data

value - the value to write.

Returns

a reference to this buffer

Exceptions

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

6.146.3.32 `virtual ByteArrayAdapter& de-
caf::internal::util::ByteArrayAdapter::putDouble
(std::size_t index, double value) throw (
lang::exceptions::IndexOutOfBoundsException) [virtual]`

Writes eight bytes containing the given value, into this buffer at the given index.

The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof(type)` if the actual start index of the type to be read.

Parameters

index - position in the Buffer to write the data

value - the value to write.

Returns

a reference to this buffer

Exceptions

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

6.146.3.33 `virtual ByteArrayAdapter& de-
caf::internal::util::ByteArrayAdapter::putDoubleAt
(std::size_t index, double value) throw (
lang::exceptions::IndexOutOfBoundsException) [virtual]`

Writes eight bytes containing the given value, into this buffer at the given byte index.

Parameters

index - position in the Buffer to write the data

value - the value to write.

Returns

a reference to this buffer

Exceptions

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

6.146.3.34 `virtual ByteArrayAdapter& de-
caf::internal::util::ByteArrayAdapter::putFloat (std::size_t index,
float value) throw (lang::exceptions::IndexOutOfBoundsException)
[virtual]`

Writes four bytes containing the given value, into this buffer at the given index.

The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof(type)` if the actual start index of the type to be read.

Parameters

index - position in the Buffer to write the data
value - the value to write.

Returns

a reference to this buffer

Exceptions

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

6.146.3.35 `virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putFloatAt (std::size_t index, float value) throw (lang::exceptions::IndexOutOfBoundsException) [virtual]`

Writes four bytes containing the given value, into this buffer at the given byte index.

Parameters

index - position in the Buffer to write the data
value - the value to write.

Returns

a reference to this buffer

Exceptions

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

6.146.3.36 `virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putInt (std::size_t index, int value) throw (lang::exceptions::IndexOutOfBoundsException) [virtual]`

Writes four bytes containing the given value, into this buffer at the given index.

The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof(type)` if the actual start index of the type to be read.

Parameters

index - position in the Buffer to write the data
value - the value to write.

Returns

a reference to this buffer

Exceptions

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

6.146.3.37 `virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putIntAt (std::size_t index, int value) throw (lang::exceptions::IndexOutOfBoundsException) [virtual]`

Writes four bytes containing the given value, into this buffer at the given byte index.

Parameters

index - position in the Buffer to write the data
value - the value to write.

Returns

a reference to this buffer

Exceptions

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

6.146.3.38 `virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putLong (std::size_t index, long long value) throw (lang::exceptions::IndexOutOfBoundsException) [virtual]`

Writes eight bytes containing the given value, into this buffer at the given index.

The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof(type)` if the actual start index of the type to be read.

Parameters

index - position in the Buffer to write the data
value - the value to write.

Returns

a reference to this buffer

Exceptions

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

6.146.3.39 `virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putLongAt (std::size_t index, long long value) throw (lang::exceptions::IndexOutOfBoundsException) [virtual]`

Writes eight bytes containing the given value, into this buffer at the given byte index.

Parameters

index - position in the Buffer to write the data

value - the value to write.

Returns

a reference to this buffer

Exceptions

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

6.146.3.40 virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putShort (std::size_t *index*, short *value*) throw (lang::exceptions::IndexOutOfBoundsException) [virtual]

Writes two bytes containing the given value, into this buffer at the given index.

The index is a relative to the size of the type to be read, in other words when accessing the element in the array `index * sizeof(type)` if the actual start index of the type to be read.

Parameters

index - position in the Buffer to write the data

value - the value to write.

Returns

a reference to this buffer

Exceptions

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

6.146.3.41 virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putShortAt (std::size_t *index*, short *value*) throw (lang::exceptions::IndexOutOfBoundsException) [virtual]

Writes two bytes containing the given value, into this buffer at the given byte index.

Parameters

index - position in the Buffer to write the data

value - the value to write.

Returns

a reference to this buffer

Exceptions

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

6.146.3.42 `virtual void decaf::internal::util::ByteArrayAdapter::read (unsigned char * buffer, std::size_t offset, std::size_t length) const throw (decaf::lang::exceptions::NullPointerException, decaf::nio::BufferUnderflowException) [virtual]`

Reads from the Byte array starting at the specified offset and reading the specified length.

If the length is greater than the capacity of this underlying byte array then an BufferUnderflowException is thrown.

Parameters

buffer - the buffer to read data from this array into.

offset - position in this array to start reading from.

length - the amount of data to read from this array.

Exceptions

NullPointerException if buffer is null

BufferUnderflowException if there is not enough data to read because the offset or the length is greater than the size of this array.

6.146.3.43 `virtual void decaf::internal::util::ByteArrayAdapter::resize (std::size_t capacity) throw (lang::exceptions::InvalidStateException) [virtual]`

Resizes the underlying array to the new given capacity, preserving all the Data that was previously in the array, unless the resize is smaller than the current size in which case only the data that will fit into the new array is preserved.

A **ByteArrayAdapter** (p. 784) can only be resized when it owns the underlying array, if it does not then it will throw an *InvalidStateException*.

Parameters

capacity - the new capacity of the array.

Exceptions

InvalidStateException if this object does not own the buffer.

6.146.3.44 `virtual void decaf::internal::util::ByteArrayAdapter::write (unsigned char * buffer, std::size_t offset, std::size_t length) throw (decaf::lang::exceptions::NullPointerException, decaf::nio::BufferOverflowException) [virtual]`

Writes from the Byte array given, starting at the specified offset and writing the specified amount of data into this objects internal array.

. If the length is greater than the capacity of this underlying byte array then an *BufferOverflowException* is thrown.

Parameters

buffer - the buffer to write get data written into this array.

offset - position in this array to start writing from.

length - the amount of data to write to this array.

Exceptions

NullPointerException if buffer is null

BufferOverflowException if the amount of data to be written to this array or the offset given are larger than this array's capacity.

The documentation for this class was generated from the following file:

- src/main/decaf/internal/util/ByteBufferAdapter.h

6.147 decaf::internal::nio::ByteBuffer Class Reference

This class defines six categories of operations upon byte buffers:

```
#include <src/main/decaf/internal/nio/ByteBuffer.h>
```

Inheritance diagram for decaf::internal::nio::ByteBuffer:

Public Member Functions

- **ByteBuffer** (std::size_t capacity, bool readOnly=false)
*Creates a **ByteBuffer** (p. 806) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*
- **ByteBuffer** (unsigned char *array, std::size_t offset, std::size_t capacity, bool readOnly=false) throw (decaf::lang::exceptions::NullPointerException)
*Creates a **ByteBuffer** (p. 806) object that wraps the given array.*
- **ByteBuffer** (ByteBufferPerspective &array, std::size_t offset, std::size_t length, bool readOnly=false) throw (decaf::lang::exceptions::IndexOutOfBoundsException)
*Creates a byte buffer that wraps the passed **ByteBufferPerspective** (p. 843) and start at the given offset.*
- **ByteBuffer** (const ByteBuffer &other)
*Create a **ByteBuffer** (p. 806) that mirrors this one, meaning it shares a reference to this buffers **ByteBufferPerspective** (p. 843) and when changes are made to that data it is reflected in both.*
- virtual ~ByteBuffer ()
- virtual bool isReadOnly () const
Tells whether or not this buffer is read-only.
- virtual unsigned char * array () throw (decaf::nio::ReadOnlyBufferException, decaf::lang::exceptions::UnsupportedOperationException)
Returns the byte array that backs this buffer.

- virtual `std::size_t arrayOffset () const throw (decaf::nio::ReadOnlyBufferException, lang::exceptions::UnsupportedOperationException)`
Returns the offset within this buffer's backing array of the first element of the buffer.
- virtual `bool hasArray () const`
Tells whether or not this buffer is backed by an accessible byte array.
- virtual `decaf::nio::CharBuffer * asCharBuffer () const`
Creates a view of this byte buffer as a char buffer.
- virtual `decaf::nio::DoubleBuffer * asDoubleBuffer () const`
Creates a view of this byte buffer as a double buffer.
- virtual `decaf::nio::FloatBuffer * asFloatBuffer () const`
Creates a view of this byte buffer as a float buffer.
- virtual `decaf::nio::IntBuffer * asIntBuffer () const`
Creates a view of this byte buffer as a int buffer.
- virtual `decaf::nio::LongBuffer * asLongBuffer () const`
Creates a view of this byte buffer as a long buffer.
- virtual `decaf::nio::ShortBuffer * asShortBuffer () const`
Creates a view of this byte buffer as a short buffer.
- virtual `ByteBuffer * asReadOnlyBuffer () const`
Creates a new, read-only byte buffer that shares this buffer's content.
- virtual `ByteBuffer & compact () throw (decaf::nio::ReadOnlyBufferException)`
Compacts this buffer.
- virtual `ByteBuffer * duplicate ()`
Creates a new byte buffer that shares this buffer's content.
- virtual `unsigned char get () const throw (decaf::nio::BufferUnderflowException)`
Relative get method.
- virtual `unsigned char get (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)`
Absolute get method.
- virtual `char getChar () throw (decaf::nio::BufferUnderflowException)`
Reads the next byte at this buffer's current position, and then increments the position by one.
- virtual `char getChar (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)`
Reads one byte at the given index and returns it.
- virtual `double getDouble () throw (decaf::nio::BufferUnderflowException)`

Reads the next eight bytes at this buffer's current position, and then increments the position by that amount.

- virtual double **getDouble** (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)

Reads eight bytes at the given index and returns it.

- virtual float **getFloat** () throw (decaf::nio::BufferUnderflowException)

Reads the next four bytes at this buffer's current position, and then increments the position by that amount.

- virtual float **getFloat** (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)

Reads four bytes at the given index and returns it.

- virtual long long **getLong** () throw (decaf::nio::BufferUnderflowException)

Reads the next eight bytes at this buffer's current position, and then increments the position by that amount.

- virtual long long **getLong** (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)

Reads eight bytes at the given index and returns it.

- virtual int **getInt** () throw (decaf::nio::BufferUnderflowException)

Reads the next four bytes at this buffer's current position, and then increments the position by that amount.

- virtual int **getInt** (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)

Reads four bytes at the given index and returns it.

- virtual short **getShort** () throw (decaf::nio::BufferUnderflowException)

Reads the next two bytes at this buffer's current position, and then increments the position by that amount.

- virtual short **getShort** (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)

Reads two bytes at the given index and returns it.

- virtual **ByteBuffer** & **put** (unsigned char value) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException)

Writes the given byte into this buffer at the current position, and then increments the position.

- virtual **ByteBuffer** & **put** (std::size_t index, unsigned char value) throw (lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException)

Writes the given byte into this buffer at the given index.

- virtual **ByteBuffer** & **putChar** (char value) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException)

Writes one byte containing the given value, into this buffer at the current position, and then increments the position by one.

- virtual **ByteBuffer** & **putChar** (std::size_t index, char value) throw (lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException)
Writes one byte containing the given value, into this buffer at the given index.
- virtual **ByteBuffer** & **putDouble** (double value) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException)
Writes eight bytes containing the given value, into this buffer at the current position, and then increments the position by eight.
- virtual **ByteBuffer** & **putDouble** (std::size_t index, double value) throw (lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException)
Writes eight bytes containing the given value, into this buffer at the given index.
- virtual **ByteBuffer** & **putFloat** (float value) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException)
Writes four bytes containing the given value, into this buffer at the current position, and then increments the position by eight.
- virtual **ByteBuffer** & **putFloat** (std::size_t index, float value) throw (lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException)
Writes four bytes containing the given value, into this buffer at the given index.
- virtual **ByteBuffer** & **putLong** (long long value) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException)
Writes eight bytes containing the given value, into this buffer at the current position, and then increments the position by eight.
- virtual **ByteBuffer** & **putLong** (std::size_t index, long long value) throw (lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException)
Writes eight bytes containing the given value, into this buffer at the given index.
- virtual **ByteBuffer** & **putInt** (int value) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException)
Writes four bytes containing the given value, into this buffer at the current position, and then increments the position by eight.
- virtual **ByteBuffer** & **putInt** (std::size_t index, int value) throw (lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException)
Writes four bytes containing the given value, into this buffer at the given index.
- virtual **ByteBuffer** & **putShort** (short value) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException)
Writes two bytes containing the given value, into this buffer at the current position, and then increments the position by eight.
- virtual **ByteBuffer** & **putShort** (std::size_t index, short value) throw (lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException)
Writes two bytes containing the given value, into this buffer at the given index.
- virtual **ByteBuffer** * **slice** () const

Creates a new byte buffer whose content is a shared subsequence of this buffer's content.

Protected Member Functions

- virtual void **setReadOnly** (bool value)

*Sets this **ByteBuffer** (p. 806) as Read-Only.*

6.147.1 Detailed Description

This class defines six categories of operations upon byte buffers: 1. Absolute and relative get and put methods that read and write single bytes; 2. Relative bulk get methods that transfer contiguous sequences of bytes from this buffer into an array; 3. Relative bulk put methods that transfer contiguous sequences of bytes from a byte array or some other byte buffer into this buffer; 4. Absolute and relative get and put methods that read and write values of other primitive types, translating them to and from sequences of bytes in a particular byte order; 5. Methods for creating view buffers, which allow a byte buffer to be viewed as a buffer containing values of some other primitive type; and 6. Methods for compacting, duplicating, and slicing a byte buffer.

Byte buffers can be created either by allocation, which allocates space for the buffer's content, or by wrapping an existing byte array into a buffer.

Access to binary data:

This class defines methods for reading and writing values of all other primitive types, except boolean. Primitive values are translated to (or from) sequences of bytes according to the buffer's current byte order.

For access to heterogeneous binary data, that is, sequences of values of different types, this class defines a family of absolute and relative get and put methods for each type. For 32-bit floating-point values, for example, this class defines:

float **getFloat()** (p. 818) float getFloat(int index) void **putFloat(float f)** (p. 824) void putFloat(int index, float f)

Corresponding methods are defined for the types char, short, int, long, and double. The index parameters of the absolute get and put methods are in terms of bytes rather than of the type being read or written.

For access to homogeneous binary data, that is, sequences of values of the same type, this class defines methods that can create views of a given byte buffer. A view buffer is simply another buffer whose content is backed by the byte buffer. Changes to the byte buffer's content will be visible in the view buffer, and vice versa; the two buffers' position, limit, and mark values are independent. The asFloatBuffer method, for example, creates an instance of the FloatBuffer class that is backed by the byte buffer upon which the method is invoked. Corresponding view-creation methods are defined for the types char, short, int, long, and double.

View buffers have two important advantages over the families of type-specific get and put methods described above:

A view buffer is indexed not in terms of bytes but rather in terms of the type-specific size of its values;

A view buffer provides relative bulk get and put methods that can transfer contiguous sequences of values between a buffer and an array or some other buffer of the same type; and

6.147.2 Constructor & Destructor Documentation

6.147.2.1 `decaf::internal::nio::ByteBuffer::ByteBuffer (std::size_t capacity, bool readOnly = false)`

Creates a **ByteBuffer** (p. 806) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.

The array is initially created with all elements initialized to zero.

Parameters

capacity - size of the array, this is the limit we read and write to.

readOnly - should this buffer be read-only, default as false

6.147.2.2 `decaf::internal::nio::ByteBuffer::ByteBuffer (unsigned char * array, std::size_t offset, std::size_t capacity, bool readOnly = false) throw (decaf::lang::exceptions::NullPointerException)`

Creates a **ByteBuffer** (p. 806) object that wraps the given array.

If the own flag is set then it will delete this array when this object is deleted.

Parameters

array - array to wrap

offset - the position that is this buffers start pos.

capacity - size of the array, this is the limit we read and write to.

readOnly - should this buffer be read-only, default as false

Exceptions

NullPointerException if buffer is NULL

6.147.2.3 `decaf::internal::nio::ByteBuffer::ByteBuffer (ByteArrayPerspective & array, std::size_t offset, std::size_t length, bool readOnly = false) throw (decaf::lang::exceptions::IndexOutOfBoundsException)`

Creates a byte buffer that wraps the passed **ByteArrayPerspective** (p. 843) and start at the given offset.

The capacity and limit of the new **ByteBuffer** (p. 806) will be that of the remaining capacity of the passed buffer.

Parameters

array - the **ByteArrayPerspective** (p. 843) to wrap

offset - the offset into array where the buffer starts

length - the length of the array we are wrapping or limit.

readOnly - is this a readOnly buffer.

Exceptions

IndexOutOfBoundsException if offset is greater than array capacity.

6.147.2.4 decaf::internal::nio::ByteBuffer::ByteBuffer (const ByteBuffer & *other*)

Create a **ByteBuffer** (p. 806) that mirrors this one, meaning it shares a reference to this buffers **ByteBufferPerspective** (p. 843) and when changes are made to that data it is reflected in both.

Parameters

other - the **ByteBuffer** (p. 806) this one is to mirror.

6.147.2.5 virtual decaf::internal::nio::ByteBuffer::~~ByteBuffer () [virtual]

6.147.3 Member Function Documentation

6.147.3.1 virtual unsigned char* decaf::internal::nio::ByteBuffer::array () throw (decaf::nio::ReadOnlyBufferException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]

Returns the byte array that backs this buffer.

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

The array that backs this buffer

Exceptions

ReadOnlyBufferException - If this buffer is backed by an array but is read-only

UnsupportedOperationException - If this buffer is not backed by an accessible array

Implements **decaf::nio::ByteBuffer** (p. 854).

6.147.3.2 virtual std::size_t decaf::internal::nio::ByteBuffer::arrayOffset () const throw (decaf::nio::ReadOnlyBufferException, lang::exceptions::UnsupportedOperationException) [virtual]

Returns the offset within this buffer's backing array of the first element of the buffer.

If this buffer is backed by an array then buffer position `p` corresponds to array index `p + arrayOffset()` (p. 812).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

The offset within this buffer's array of the first element of the buffer

Exceptions

ReadOnlyBufferException - If this buffer is backed by an array but is read-only

UnsupportedOperationException - If this buffer is not backed by an accessible array

Implements **decaf::nio::ByteBuffer** (p. 854).

6.147.3.3 `virtual decaf::nio::CharBuffer* decaf::internal::nio::ByteArrayBuffer::asCharBuffer ()`
`const [inline, virtual]`

Creates a view of this byte buffer as a char buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the new Char Buffer, which the caller then owns.

Implements **decaf::nio::ByteBuffer** (p. 855).

6.147.3.4 `virtual decaf::nio::DoubleBuffer* decaf::internal::nio::ByteArrayBuffer::asDoubleBuffer ()`
`const [inline, virtual]`

Creates a view of this byte buffer as a double buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by eight, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the new double Buffer, which the caller then owns.

Implements **decaf::nio::ByteBuffer** (p. 855).

6.147.3.5 `virtual decaf::nio::FloatBuffer* decaf::internal::nio::ByteArrayBuffer::asFloatBuffer ()`
`const [inline, virtual]`

Creates a view of this byte buffer as a float buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by four, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the new float Buffer, which the caller then owns.

Implements **decaf::nio::ByteBuffer** (p. 855).

6.147.3.6 `virtual decaf::nio::IntBuffer* decaf::internal::nio::ByteBuffer::asIntBuffer () const [inline, virtual]`

Creates a view of this byte buffer as a int buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by four, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the new int Buffer, which the caller then owns.

Implements **decaf::nio::ByteBuffer** (p. 856).

6.147.3.7 `virtual decaf::nio::LongBuffer* decaf::internal::nio::ByteBuffer::asLongBuffer () const [inline, virtual]`

Creates a view of this byte buffer as a long buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by eight, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the new long Buffer, which the caller then owns.

Implements **decaf::nio::ByteBuffer** (p. 856).

6.147.3.8 `virtual ByteBuffer* decaf::internal::nio::ByteBuffer::asReadOnlyBuffer () const [virtual]`

Creates a new, read-only byte buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns

The new, read-only byte buffer which the caller then owns.

Implements **decaf::nio::ByteBuffer** (p. 856).

6.147.3.9 **virtual decaf::nio::ShortBuffer* decaf::internal::nio::ByteBuffer::asShortBuffer ()**
const [inline, virtual]

Creates a view of this byte buffer as a short buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by two, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the new short Buffer, which the caller then owns.

Implements **decaf::nio::ByteBuffer** (p. 857).

6.147.3.10 **virtual ByteBuffer& decaf::internal::nio::ByteBuffer::compact () throw (decaf::nio::ReadOnlyBufferException) [virtual]**

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index `p = position()` (p. 746) is copied to index zero, the byte at index `p + 1` is copied to index one, and so forth until the byte at index `limit()` (p. 745) - 1 is copied to index `n = limit() (p. 745) - 1 - p`. The buffer's position is then set to `n+1` and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns

a reference to this **ByteBuffer** (p. 806)

Exceptions

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::ByteBuffer** (p. 857).

6.147.3.11 `virtual ByteBuffer* decaf::internal::nio::ByteBuffer::duplicate ()`
[virtual]

Creates a new byte buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

a new Byte Buffer which the caller owns.

Implements **decaf::nio::ByteBuffer** (p. 858).

6.147.3.12 `virtual unsigned char decaf::internal::nio::ByteBuffer::get ()`
`const throw (decaf::nio::BufferUnderflowException)` [virtual]

Relative get method.

Reads the byte at this buffer's current position, and then increments the position.

Returns

The byte at the buffer's current position

Exceptions

BufferUnderflowException - If the buffer's current position is not smaller than its limit

Implements **decaf::nio::ByteBuffer** (p. 859).

6.147.3.13 `virtual unsigned char decaf::internal::nio::ByteBuffer::get`
`(std::size_t index) const throw (`
`lang::exceptions::IndexOutOfBoundsException)`
[virtual]

Absolute get method.

Reads the byte at the given index.

Parameters

index - the index in the Buffer where the byte is to be read

Returns

the byte that is located at the given index

Exceptions

IndexOutOfBoundsException - If index is not smaller than the buffer's limit

Implements **decaf::nio::ByteBuffer** (p. 859).

6.147.3.14 `virtual char decaf::internal::nio::ByteBuffer::getChar () throw (decaf::nio::BufferUnderflowException) [inline, virtual]`

Reads the next byte at this buffer's current position, and then increments the position by one.

Returns

the next char in the buffer..

Exceptions

BufferUnderflowException - If there are no more bytes remaining in this buffer, meaning we have reached the set limit.

Implements **decaf::nio::ByteBuffer** (p. 860).

6.147.3.15 `virtual char decaf::internal::nio::ByteBuffer::getChar (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException) [inline, virtual]`

Reads one byte at the given index and returns it.

Parameters

index - the index in the Buffer where the byte is to be read

Returns

the char at the given index in the buffer

Exceptions

IndexOutOfBoundsException - If index is not smaller than the buffer's limit

Implements **decaf::nio::ByteBuffer** (p. 860).

6.147.3.16 `virtual double decaf::internal::nio::ByteBuffer::getDouble () throw (decaf::nio::BufferUnderflowException) [virtual]`

Reads the next eight bytes at this buffer's current position, and then increments the position by that amount.

Returns

the next double in the buffer..

Exceptions

BufferUnderflowException - If there are no more bytes remaining in this buffer, meaning we have reached the set limit.

Implements **decaf::nio::ByteBuffer** (p. 861).

6.147.3.17 virtual double decaf::internal::nio::ByteBuffer::getDouble
(std::size_t *index*) const throw (
lang::exceptions::IndexOutOfBoundsException)
[virtual]

Reads eight bytes at the given index and returns it.

Parameters

index - the index in the Buffer where the bytes are to be read

Returns

the double at the given index in the buffer

Exceptions

IndexOutOfBoundsException - If there are not enough bytes remaining to fill the requested Data Type

Implements **decaf::nio::ByteBuffer** (p. 861).

6.147.3.18 virtual float decaf::internal::nio::ByteBuffer::getFloat () throw (
decaf::nio::BufferUnderflowException) [virtual]

Reads the next four bytes at this buffer's current position, and then increments the position by that amount.

Returns

the next float in the buffer..

Exceptions

BufferUnderflowException - If there are no more bytes remaining in this buffer, meaning we have reached the set limit.

Implements **decaf::nio::ByteBuffer** (p. 861).

6.147.3.19 virtual float decaf::internal::nio::ByteBuffer::getFloat (std::size_t *index*) const throw (lang::exceptions::IndexOutOfBoundsException)
[virtual]

Reads four bytes at the given index and returns it.

Parameters

index - the index in the Buffer where the bytes are to be read

Returns

the float at the given index in the buffer

Exceptions

IndexOutOfBoundsException - If there are not enough bytes remaining to fill the requested Data Type

Implements **decaf::nio::ByteBuffer** (p. 862).

6.147.3.20 `virtual int decaf::internal::nio::ByteArrayBuffer::getInt () throw (decaf::nio::BufferUnderflowException) [virtual]`

Reads the next four bytes at this buffer's current position, and then increments the position by that amount.

Returns

the next int in the buffer..

Exceptions

BufferUnderflowException - If there are no more bytes remaining in this buffer, meaning we have reached the set limit.

Implements **decaf::nio::ByteBuffer** (p. 862).

6.147.3.21 `virtual int decaf::internal::nio::ByteArrayBuffer::getInt (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException) [virtual]`

Reads four bytes at the given index and returns it.

Parameters

index - the index in the Buffer where the bytes are to be read

Returns

the int at the given index in the buffer

Exceptions

IndexOutOfBoundsException - If there are not enough bytes remaining to fill the requested Data Type

Implements **decaf::nio::ByteBuffer** (p. 862).

6.147.3.22 `virtual long long decaf::internal::nio::ByteArrayBuffer::getLong () throw (decaf::nio::BufferUnderflowException) [virtual]`

Reads the next eight bytes at this buffer's current position, and then increments the position by that amount.

Returns

the next long long in the buffer..

Exceptions

BufferUnderflowException - If there are no more bytes remaining in this buffer, meaning we have reached the set limit.

Implements **decaf::nio::ByteBuffer** (p. 863).

6.147.3.23 `virtual long long decaf::internal::nio::ByteBuffer::getLong
 (std::size_t index) const throw (
 lang::exceptions::IndexOutOfBoundsException)
 [virtual]`

Reads eight bytes at the given index and returns it.

Parameters

index - the index in the Buffer where the bytes are to be read

Returns

the long long at the given index in the buffer

Exceptions

IndexOutOfBoundsException - If there are not enough bytes remaining to fill the requested Data Type

Implements **decaf::nio::ByteBuffer** (p. 863).

6.147.3.24 `virtual short decaf::internal::nio::ByteBuffer::getShort (std::size_t
 index) const throw (lang::exceptions::IndexOutOfBoundsException
) [virtual]`

Reads two bytes at the given index and returns it.

Parameters

index - the index in the Buffer where the bytes are to be read

Returns

the short at the given index in the buffer

Exceptions

IndexOutOfBoundsException - If there are not enough bytes remaining to fill the requested Data Type

Implements **decaf::nio::ByteBuffer** (p. 864).

6.147.3.25 `virtual short decaf::internal::nio::ByteBuffer::getShort () throw
 (decaf::nio::BufferUnderflowException) [virtual]`

Reads the next two bytes at this buffer's current position, and then increments the position by that amount.

Returns

the next short in the buffer..

Exceptions

BufferUnderflowException - If there are no more bytes remaining in this buffer, meaning we have reached the set limit.

Implements **decaf::nio::ByteBuffer** (p. 863).

6.147.3.26 `virtual bool decaf::internal::nio::ByteArrayBuffer::hasArray () const`
`[inline, virtual]`

Tells whether or not this buffer is backed by an accessible byte array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Sub-classes should override this method if they do not have a backing array as this class always returns true.

Returns

true if, and only if, this buffer is backed by an array and is not read-only

Implements `decaf::nio::ByteBuffer` (p. 864).

6.147.3.27 `virtual bool decaf::internal::nio::ByteArrayBuffer::isReadOnly ()`
`const [inline, virtual]`

Tells whether or not this buffer is read-only.

Returns

true if, and only if, this buffer is read-only

Implements `decaf::nio::ByteBuffer` (p. 864).

6.147.3.28 `virtual ByteArrayBuffer& decaf::internal::nio::ByteArrayBuffer::put (`
`unsigned char value) throw (decaf::nio::BufferOverflowException,`
`decaf::nio::ReadOnlyBufferException) [virtual]`

Writes the given byte into this buffer at the current position, and then increments the position.

Parameters

value - the byte value to be written

Returns

a reference to this buffer

Exceptions

BufferOverflowException - If this buffer's current position is not smaller than its limit

ReadOnlyBufferException - If this buffer is read-only

Implements `decaf::nio::ByteBuffer` (p. 865).

6.147.3.29 `virtual ByteArrayBuffer& decaf::internal::nio::ByteArrayBuffer::put`
`(std::size_t index, unsigned char value) throw`
`(lang::exceptions::IndexOutOfBoundsException,`
`decaf::nio::ReadOnlyBufferException) [virtual]`

Writes the given byte into this buffer at the given index.

Parameters

index - position in the Buffer to write the data
value - the byte to write.

Returns

a reference to this buffer

Exceptions

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::ByteBuffer** (p. 865).

6.147.3.30 `virtual ByteBuffer& decaf::internal::nio::ByteBuffer::putChar (std::size_t index, char value) throw (lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException) [virtual]`

Writes one byte containing the given value, into this buffer at the given index.

Parameters

index - position in the Buffer to write the data
value - the value to write.

Returns

a reference to this buffer

Exceptions

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::ByteBuffer** (p. 868).

6.147.3.31 `virtual ByteBuffer& decaf::internal::nio::ByteBuffer::putChar (char value) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException) [virtual]`

Writes one byte containing the given value, into this buffer at the current position, and then increments the position by one.

Parameters

value - The value to be written

Returns

a reference to this buffer

Exceptions

BufferOverflowException - If there are fewer than bytes remaining in this buffer than the size of the data to be written

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::ByteBuffer** (p. 867).

6.147.3.32 **virtual ByteBuffer& decaf::internal::nio::ByteBuffer::putDouble (std::size_t index, double value) throw (lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException) [virtual]**

Writes eight bytes containing the given value, into this buffer at the given index.

Parameters

index - position in the Buffer to write the data

value - the value to write.

Returns

a reference to this buffer

Exceptions

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::ByteBuffer** (p. 869).

6.147.3.33 **virtual ByteBuffer& decaf::internal::nio::ByteBuffer::putDouble (double value) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException) [virtual]**

Writes eight bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

Parameters

value - The value to be written

Returns

a reference to this buffer

Exceptions

BufferOverflowException - If there are fewer than bytes remaining in this buffer than the size of the data to be written

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::ByteBuffer** (p. 868).

6.147.3.34 virtual ByteBuffer& decaf::internal::nio::ByteBuffer::putFloat (std::size_t *index*, float *value*) throw (lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException) [virtual]

Writes four bytes containing the given value, into this buffer at the given index.

Parameters

index - position in the Buffer to write the data
value - the value to write.

Returns

a reference to this buffer

Exceptions

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::ByteBuffer** (p. 869).

6.147.3.35 virtual ByteBuffer& decaf::internal::nio::ByteBuffer::putFloat (float *value*) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException) [virtual]

Writes four bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

Parameters

value - The value to be written

Returns

a reference to this buffer

Exceptions

BufferOverflowException - If there are fewer than bytes remaining in this buffer than the size of the data to be written

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::ByteBuffer** (p. 869).

6.147.3.36 virtual ByteBuffer& decaf::internal::nio::ByteBuffer::putInt (std::size_t *index*, int *value*) throw (lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException) [virtual]

Writes four bytes containing the given value, into this buffer at the given index.

Parameters

index - position in the Buffer to write the data
value - the value to write.

Returns

a reference to this buffer

Exceptions

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::ByteBuffer** (p. 870).

6.147.3.37 `virtual ByteBuffer& decaf::internal::nio::ByteBuffer::putInt (int value) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException) [virtual]`

Writes four bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

Parameters

value - The value to be written

Returns

a reference to this buffer

Exceptions

BufferOverflowException - If there are fewer than bytes remaining in this buffer than the size of the data to be written

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::ByteBuffer** (p. 870).

6.147.3.38 `virtual ByteBuffer& decaf::internal::nio::ByteBuffer::putLong (long long value) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException) [virtual]`

Writes eight bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

Parameters

value - The value to be written

Returns

a reference to this buffer

Exceptions

BufferOverflowException - If there are fewer than bytes remaining in this buffer than the size of the data to be written

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::ByteBuffer** (p. 871).

6.147.3.39 **virtual ByteBuffer& decaf::internal::nio::ByteBuffer::putLong (std::size_t index, long long value) throw (lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException) [virtual]**

Writes eight bytes containing the given value, into this buffer at the given index.

Parameters

index - position in the Buffer to write the data

value - the value to write.

Returns

a reference to this buffer

Exceptions

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::ByteBuffer** (p. 871).

6.147.3.40 **virtual ByteBuffer& decaf::internal::nio::ByteBuffer::putShort (short value) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException) [virtual]**

Writes two bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

Parameters

value - The value to be written

Returns

a reference to this buffer

Exceptions

BufferOverflowException - If there are fewer than bytes remaining in this buffer than the size of the data to be written

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::ByteBuffer** (p. 872).

6.147.3.41 `virtual ByteBuffer& decaf::internal::nio::ByteBuffer::putShort (std::size_t index, short value) throw (lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException) [virtual]`

Writes two bytes containing the given value, into this buffer at the given index.

Parameters

index - position in the Buffer to write the data
value - the value to write.

Returns

a reference to this buffer

Exceptions

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::ByteBuffer** (p. 872).

6.147.3.42 `virtual void decaf::internal::nio::ByteBuffer::setReadOnly (bool value) [inline, protected, virtual]`

Sets this **ByteBuffer** (p. 806) as Read-Only.

Parameters

value - true if this buffer is to be read-only.

6.147.3.43 `virtual ByteBuffer* decaf::internal::nio::ByteBuffer::slice () const [virtual]`

Creates a new byte buffer whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the newly create **ByteBuffer** (p. 806) which the caller owns.

Implements **decaf::nio::ByteBuffer** (p. 872).

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/nio/ByteBuffer.h`

6.148 decaf::io::ByteArrayInputStream Class Reference

Simple implementation of **InputStream** (p.1630) that wraps around an STL Vector `std::vector<unsigned char>`.

```
#include <src/main/decaf/io/ByteArrayInputStream.h>
```

Inheritance diagram for `decaf::io::ByteArrayInputStream`:

Public Member Functions

- **ByteArrayInputStream** ()
Default Constructor.
- **ByteArrayInputStream** (const std::vector< unsigned char > &buffer)
Creates the input stream and calls setBuffer with the specified buffer object.
- **ByteArrayInputStream** (const unsigned char *buffer, std::size_t bufferSize)
Constructor.
- virtual ~**ByteArrayInputStream** ()
- virtual void **setBuffer** (const std::vector< unsigned char > &buffer)
Sets the internal buffer.
- virtual void **setByteArray** (const unsigned char *buffer, std::size_t bufferSize)
Sets the data that this reader uses, replaces any existing data and resets to beginning of the buffer.
- virtual std::size_t **available** () const throw (IOException)
Indicates the number of bytes available.
- virtual int **read** () throw (IOException)
Reads a single byte from the buffer.
- virtual int **read** (unsigned char *buffer, std::size_t offset, std::size_t bufferSize) throw (IOException, lang::exceptions::NullPointerException)
Reads an array of bytes from the buffer.
- virtual void **close** () throw (io::IOException)
Closes the target input stream.
- virtual std::size_t **skip** (std::size_t num) throw (io::IOException, lang::exceptions::UnsupportedOperationException)
Skips over and discards n bytes of data from this input stream.
- virtual void **mark** (int readLimit DECAF_UNUSED)
Marks the current position in the stream A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.
- virtual void **reset** () throw (IOException)

Resets the read index to the beginning of the byte array, unless mark has been called and the markLimit has not been exceeded, in which case the stream is reset to the marked position.

- virtual bool **markSupported** () const

Determines if this input stream supports the mark and reset methods.

Protected Member Functions

- virtual void **lock** () throw (decaf::lang::exceptions::RuntimeException)

Locks the object.

- virtual bool **tryLock** () throw (decaf::lang::exceptions::RuntimeException)

Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

- virtual void **unlock** () throw (decaf::lang::exceptions::RuntimeException)

Unlocks the object.

- virtual void **wait** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)

Waits on a signal from this object, which is generated by a call to Notify.

- virtual void **wait** (long long millisecs) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)

Waits on a signal from this object, which is generated by a call to Notify.

- virtual void **wait** (long long millisecs, int nanos) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)

Waits on a signal from this object, which is generated by a call to Notify.

- virtual void **notify** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)

Signals a waiter on this object that it can now wake up and continue.

- virtual void **notifyAll** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)

Signals the waiters on this object that it can now wake up and continue.

6.148.1 Detailed Description

Simple implementation of **InputStream** (p.1630) that wraps around an STL Vector `std::vector<unsigned char>`. Closing a **ByteArrayInputStream** (p.828) has no effect. The methods in this class can be called after the stream has been closed without generating an **IOException** (p.1707).

6.148.2 Constructor & Destructor Documentation

6.148.2.1 decaf::io::ByteArrayInputStream::ByteArrayInputStream ()

Default Constructor.

6.148.2.2 decaf::io::ByteArrayInputStream::ByteArrayInputStream (const std::vector< unsigned char > & *buffer*)

Creates the input stream and calls setBuffer with the specified buffer object.

Parameters

buffer The buffer to be used.

6.148.2.3 decaf::io::ByteArrayInputStream::ByteArrayInputStream (const unsigned char * *buffer*, std::size_t *bufferSize*)

Constructor.

Parameters

buffer initial byte array to use to read from

bufferSize the size of the buffer

6.148.2.4 virtual decaf::io::ByteArrayInputStream::~~ByteArrayInputStream () [virtual]

6.148.3 Member Function Documentation

6.148.3.1 virtual std::size_t decaf::io::ByteArrayInputStream::available () const throw (IOException) [inline, virtual]

Indicates the number of bytes available.

Returns

The number of bytes until the end of the internal buffer.

Implements **decaf::io::InputStream** (p. 1631).

6.148.3.2 virtual void decaf::io::ByteArrayInputStream::close () throw (io::IOException) [inline, virtual]

Closes the target input stream.

Exceptions

IOException (p. 1707) thrown if an error occurs.

6.148.3.3 `virtual void decaf::io::ByteArrayInputStream::lock () throw (decaf::lang::exceptions::RuntimeException) [inline, protected, virtual]`

Locks the object.

Exceptions

RuntimeException if an error occurs while locking the object.

Implements `decaf::util::concurrent::Synchronizable` (p. 2932).

6.148.3.4 `virtual void decaf::io::ByteArrayInputStream::mark (int readLimit DECAF_UNUSED) [inline, virtual]`

Marks the current position in the stream. A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.

If a stream instance reports that marks are supported then the stream will ensure that the same bytes can be read again after the reset method is called so long the readLimit is not reached.

Parameters

readLimit - max bytes read before marked position is invalid.

Implements `decaf::io::InputStream` (p. 1631).

6.148.3.5 `virtual bool decaf::io::ByteArrayInputStream::markSupported () const [inline, virtual]`

Determines if this input stream supports the mark and reset methods.

Whether or not mark and reset are supported is an invariant property of a particular input stream instance.

Returns

true if this stream instance supports marks

Implements `decaf::io::InputStream` (p. 1631).

6.148.3.6 `virtual void decaf::io::ByteArrayInputStream::notify () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException) [inline, protected, virtual]`

Signals a waiter on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

RuntimeException if an error occurs while notifying one of the waiting threads.

Implements `decaf::util::concurrent::Synchronizable` (p. 2933).

6.148.3.7 `virtual void decaf::io::ByteArrayInputStream::notifyAll () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException) [inline, protected, virtual]`

Signals the waiters on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

RuntimeException if an error occurs while notifying the waiting threads.

Implements `decaf::util::concurrent::Synchronizable` (p. 2934).

6.148.3.8 `virtual int decaf::io::ByteArrayInputStream::read () throw (IOException) [virtual]`

Reads a single byte from the buffer.

Returns

The next byte.

Exceptions

IOException (p. 1707) thrown if an error occurs.

Implements `decaf::io::InputStream` (p. 1632).

6.148.3.9 `virtual int decaf::io::ByteArrayInputStream::read (unsigned char * buffer, std::size_t offset, std::size_t bufferSize) throw (IOException, lang::exceptions::NullPointerException) [virtual]`

Reads an array of bytes from the buffer.

Parameters

buffer (out) the target buffer.

offset the position in the buffer to start reading from.

bufferSize the size of the output buffer.

Returns

The number of bytes read.

Exceptions

IOException (p. 1707) thrown if an error occurs.

Implements `decaf::io::InputStream` (p. 1632).

6.148.3.10 `virtual void decaf::io::ByteArrayInputStream::reset () throw (IOException) [virtual]`

Resets the read index to the beginning of the byte array, unless mark has been called and the markLimit has not been exceeded, in which case the stream is reset to the marked position.

Implements **decaf::io::InputStream** (p. 1633).

6.148.3.11 `virtual void decaf::io::ByteArrayInputStream::setBuffer (const std::vector< unsigned char > & buffer) [virtual]`

Sets the internal buffer.

The input stream will wrap around this buffer and will perform all read operations on it. The position will be reinitialized to the beginning of the specified buffer. This class will not own the given buffer - it is the caller's responsibility to free the memory of the given buffer as appropriate.

Parameters

buffer The buffer to be used.

6.148.3.12 `virtual void decaf::io::ByteArrayInputStream::setByteArray (const unsigned char * buffer, std::size_t bufferSize) [virtual]`

Sets the data that this reader uses, replaces any existing data and resets to beginning of the buffer.

Parameters

buffer initial byte array to use to read from

bufferSize the size of the buffer

6.148.3.13 `virtual std::size_t decaf::io::ByteArrayInputStream::skip (std::size_t num) throw (io::IOException, lang::exceptions::UnsupportedOperationException) [virtual]`

Skips over and discards n bytes of data from this input stream.

The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before n bytes have been skipped is only one possibility. The actual number of bytes skipped is returned. If n is negative, no bytes are skipped.

The skip method of **InputStream** (p. 1630) creates a byte array and then repeatedly reads into it until n bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.

Parameters

num - the number of bytes to skip

Returns

total bytes skipped

Exceptions

IOException (p. 1707) if an error occurs

Implements **decaf::io::InputStream** (p. 1633).

6.148.3.14 `virtual bool decaf::io::ByteArrayInputStream::tryLock () throw (decaf::lang::exceptions::RuntimeException) [inline, protected, virtual]`

Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

Returns

true if the lock was acquired, false if it is already held by another thread.

Exceptions

RuntimeException if an error occurs while locking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2935).

6.148.3.15 `virtual void decaf::io::ByteArrayInputStream::unlock () throw (decaf::lang::exceptions::RuntimeException) [inline, protected, virtual]`

Unlocks the object.

Exceptions

RuntimeException if an error occurs while unlocking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2936).

6.148.3.16 `virtual void decaf::io::ByteArrayInputStream::wait (long long millisecs, int nanos) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException) [inline, protected, virtual]`

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

Parameters

millisecs the time in milliseconds to wait, or WAIT_INFINITE

nanos additional time in nanoseconds with a range of 0-999999

Exceptions

IllegalArgumentException if an error occurs or the *nanos* argument is not in the range of [0-999999]

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements `decaf::util::concurrent::Synchronizable` (p. 2940).

```
6.148.3.17  virtual void decaf::io::ByteArrayInputStream::wait ( long long
               millisecs ) throw ( decaf::lang::exceptions::RuntimeException,
               decaf::lang::exceptions::IllegalMonitorStateException,
               decaf::lang::exceptions::InterruptedException ) [inline, protected,
               virtual]
```

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval.

Parameters

millisecs the time in milliseconds to wait, or WAIT_INFINITE

Exceptions

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements `decaf::util::concurrent::Synchronizable` (p. 2938).

```
6.148.3.18  virtual void decaf::io::ByteArrayInputStream::wait (      )
               throw ( decaf::lang::exceptions::RuntimeException,
               decaf::lang::exceptions::IllegalMonitorStateException,
               decaf::lang::exceptions::InterruptedException ) [inline, protected,
               virtual]
```

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling.

Exceptions

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2937).

The documentation for this class was generated from the following file:

- `src/main/decaf/io/ByteArrayInputStream.h`

6.149 decaf::io::ByteArrayOutputStream Class Reference

```
#include <src/main/decaf/io/ByteArrayOutputStream.h>
```

Inheritance diagram for decaf::io::ByteArrayOutputStream:

Public Member Functions

- **ByteArrayOutputStream** ()
Default Constructor - uses a default internal buffer.
- **ByteArrayOutputStream** (std::vector< unsigned char > &buffer)
Uses the given buffer as the target.
- virtual ~**ByteArrayOutputStream** ()
- virtual void **setBuffer** (std::vector< unsigned char > &buffer)
Sets the internal buffer.
- virtual const unsigned char * **toByteArray** () const
Get a snapshot of the data.
- virtual const std::vector< unsigned char > **toByteArrayRef** () const
Get a snapshot of the data.
- virtual std::size_t **size** () const
Get the Size of the Internal Buffer.
- virtual void **write** (unsigned char c) throw (IOException)
Writes a single byte to the output stream.
- virtual void **write** (const std::vector< unsigned char > &buffer) throw (IOException)
Writes an array of bytes to the output stream.
- virtual void **write** (const unsigned char *buffer, std::size_t offset, std::size_t len) throw (IOException, lang::exceptions::NullPointerException)
Writes an array of bytes to the output stream.
- virtual void **flush** () throw (IOException)
Invokes flush on the target output stream, has no affect.
- virtual void **reset** () throw (IOException)
Clear current Stream contents.

- `void close () throw (io::IOException)`
Invokes close on the target output stream.
- `std::string toString () const`
Converts the bytes in the buffer into a standard C++ string.
- `void writeTo (OutputStream *out) const throw (IOException, lang::exceptions::NullPointerException)`
Writes the complete contents of this byte array output stream to the specified output stream argument, as if by calling the output stream's write method using out.write(buf, 0, count).
- `virtual void lock () throw (decaf::lang::exceptions::RuntimeException)`
Locks the object.
- `virtual bool tryLock () throw (decaf::lang::exceptions::RuntimeException)`
Attempts to Lock the object, if the lock is already held by another thread than this method returns false.
- `virtual void unlock () throw (decaf::lang::exceptions::RuntimeException)`
Unlocks the object.
- `virtual void wait () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)`
Waits on a signal from this object, which is generated by a call to Notify.
- `virtual void wait (long long millisecs) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)`
Waits on a signal from this object, which is generated by a call to Notify.
- `virtual void wait (long long millisecs, int nanos) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)`
Waits on a signal from this object, which is generated by a call to Notify.
- `virtual void notify () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)`
Signals a waiter on this object that it can now wake up and continue.
- `virtual void notifyAll () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)`
Signals the waiters on this object that it can now wake up and continue.

6.149.1 Constructor & Destructor Documentation

6.149.1.1 decaf::io::ByteArrayOutputStream::ByteArrayOutputStream ()

Default Constructor - uses a default internal buffer.

6.149.1.2 decaf::io::ByteArrayOutputStream::ByteArrayOutputStream (std::vector< unsigned char > & *buffer*)

Uses the given buffer as the target.

Calls setBuffer.

Parameters

buffer the target buffer.

6.149.1.3 virtual decaf::io::ByteArrayOutputStream::~~ByteArrayOutputStream () [inline, virtual]

6.149.2 Member Function Documentation

6.149.2.1 void decaf::io::ByteArrayOutputStream::close () throw (io::IOException) [inline]

Invokes close on the target output stream.

Exceptions

IOException (p. 1707)

6.149.2.2 virtual void decaf::io::ByteArrayOutputStream::flush () throw (IOException) [inline, virtual]

Invokes flush on the target output stream, has no affect.

Exceptions

IOException (p. 1707)

Implements **decaf::io::OutputStream** (p. 2317).

6.149.2.3 virtual void decaf::io::ByteArrayOutputStream::lock () throw (decaf::lang::exceptions::RuntimeException) [inline, virtual]

Locks the object.

Exceptions

RuntimeException if an error occurs while locking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2932).

6.149.2.4 virtual void decaf::io::ByteArrayOutputStream::notify () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException) [inline, virtual]

Signals a waiter on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

RuntimeException if an error occurs while notifying one of the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 2933).

6.149.2.5 `virtual void decaf::io::ByteArrayOutputStream::notifyAll () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException) [inline, virtual]`

Signals the waiters on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

RuntimeException if an error occurs while notifying the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 2934).

6.149.2.6 `virtual void decaf::io::ByteArrayOutputStream::reset () throw (IOException) [virtual]`

Clear current Stream contents.

Exceptions

IOException (p. 1707)

6.149.2.7 `virtual void decaf::io::ByteArrayOutputStream::setBuffer (std::vector< unsigned char > & buffer) [virtual]`

Sets the internal buffer.

This input stream will wrap around the given buffer and all writes will be performed directly on the buffer. This object does not retain control of the buffer's lifetime however - this is the job of the caller.

Parameters

buffer The target buffer.

6.149.2.8 `virtual std::size_t decaf::io::ByteArrayOutputStream::size () const`
[inline, virtual]

Get the Size of the Internal Buffer.

Returns

size of the internal buffer

6.149.2.9 `virtual const unsigned char* decaf::io::ByteArrayOutputStream::toByteArray () const`
[inline, virtual]

Get a snapshot of the data.

Returns

pointer to the data

6.149.2.10 `virtual const std::vector<unsigned char> decaf::io::ByteArrayOutputStream::toByteArrayRef () const` [inline, virtual]

Get a snapshot of the data.

Returns

reference to the underlying data as a const std::vector<unsigned char>&

6.149.2.11 `std::string decaf::io::ByteArrayOutputStream::toString () const`

Converts the bytes in the buffer into a standard C++ string.

Returns

a string containing the bytes in the buffer

6.149.2.12 `virtual bool decaf::io::ByteArrayOutputStream::tryLock () throw (decaf::lang::exceptions::RuntimeException)` [inline, virtual]

Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

Returns

true if the lock was acquired, false if it is already held by another thread.

Exceptions

RuntimeException if an error occurs while locking the object.

Implements `decaf::util::concurrent::Synchronizable` (p. 2935).

6.149.2.13 `virtual void decaf::io::ByteArrayOutputStream::unlock () throw (decaf::lang::exceptions::RuntimeException) [inline, virtual]`

Unlocks the object.

Exceptions

RuntimeException if an error occurs while unlocking the object.

Implements `decaf::util::concurrent::Synchronizable` (p. 2936).

6.149.2.14 `virtual void decaf::io::ByteArrayOutputStream::wait () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling.

Exceptions

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements `decaf::util::concurrent::Synchronizable` (p. 2937).

6.149.2.15 `virtual void decaf::io::ByteArrayOutputStream::wait (long long millisecs) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval.

Parameters

millisecs the time in milliseconds to wait, or WAIT_INFINITE

Exceptions

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements `decaf::util::concurrent::Synchronizable` (p. 2938).

6.149.2.16 virtual void decaf::io::ByteArrayOutputStream::wait
 (long long *millisecs*, int *nanos*) throw
 (decaf::lang::exceptions::RuntimeException,
 decaf::lang::exceptions::IllegalArgumentException,
 decaf::lang::exceptions::IllegalMonitorStateException,
 decaf::lang::exceptions::InterruptedException) [inline, virtual]

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

Parameters

millisecs the time in milliseconds to wait, or WAIT_INFINITE

nanos additional time in nanoseconds with a range of 0-999999

Exceptions

IllegalArgumentException if an error occurs or the nanos argument is not in the range of [0-999999]

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements decaf::util::concurrent::Synchronizable (p. 2940).

6.149.2.17 virtual void decaf::io::ByteArrayOutputStream::write (unsigned char
c) throw (IOException) [virtual]

Writes a single byte to the output stream.

Parameters

c the byte.

Exceptions

IOException (p. 1707) thrown if an error occurs.

Implements decaf::io::OutputStream (p. 2318).

6.149.2.18 virtual void decaf::io::ByteArrayOutputStream::write (const unsigned
 char * *buffer*, std::size_t *offset*, std::size_t *len*) throw (
 IOException, lang::exceptions::NullPointerException) [virtual]

Writes an array of bytes to the output stream.

Parameters

- buffer* The array of bytes to write.
- offset* the position to start writing in buffer.
- len* The number of bytes from the buffer to be written.

Exceptions

- IOException* (p. 1707) thrown if an error occurs.
- NullPointerException* thrown if buffer is Null.

Implements **decaf::io::OutputStream** (p. 2317).

6.149.2.19 `virtual void decaf::io::ByteArrayOutputStream::write (const
std::vector< unsigned char > & buffer) throw (IOException)
[virtual]`

Writes an array of bytes to the output stream.

Parameters

- buffer* The bytes to write.

Exceptions

- IOException* (p. 1707) thrown if an error occurs.

Implements **decaf::io::OutputStream** (p. 2317).

6.149.2.20 `void decaf::io::ByteArrayOutputStream::writeTo (OutputStream * out
) const throw (IOException, lang::exceptions::NullPointerException)`

Writes the complete contents of this byte array output stream to the specified output stream argument, as if by calling the output stream's write method using `out.write(buf, 0, count)`.

The documentation for this class was generated from the following file:

- `src/main/decaf/io/ByteArrayOutputStream.h`

6.150 decaf::internal::nio::ByteArrayPerspective Class Reference

This class extends `ByteArray` to create a reference counted byte array that can be held and used by several different `ByteBuffers` and allow them to know on destruction whose job it is to delete the perspective.

```
#include <src/main/decaf/internal/nio/ByteArrayPerspective.h>
```

Inheritance diagram for `decaf::internal::nio::ByteArrayPerspective`:

Public Member Functions

- **ByteArrayPerspective** (std::size_t capacity)
Creates a byte array object that is allocated internally and is then owned and deleted when this object is deleted.
- **ByteArrayPerspective** (unsigned char *array, std::size_t capacity, bool own=false) throw (lang::exceptions::NullPointerException)
Creates a byte array object that wraps the given array.
- **ByteArrayPerspective** (char *array, std::size_t capacity, bool own=false) throw (lang::exceptions::NullPointerException)
Creates a array object that wraps the given array.
- **ByteArrayPerspective** (double *array, std::size_t capacity, bool own=false) throw (lang::exceptions::NullPointerException)
Creates a array object that wraps the given array.
- **ByteArrayPerspective** (float *array, std::size_t capacity, bool own=false) throw (lang::exceptions::NullPointerException)
Creates a array object that wraps the given array.
- **ByteArrayPerspective** (long long *array, std::size_t capacity, bool own=false) throw (lang::exceptions::NullPointerException)
Creates a array object that wraps the given array.
- **ByteArrayPerspective** (int *array, std::size_t capacity, bool own=false) throw (lang::exceptions::NullPointerException)
Creates a array object that wraps the given array.
- **ByteArrayPerspective** (short *array, std::size_t capacity, bool own=false) throw (lang::exceptions::NullPointerException)
Creates a array object that wraps the given array.
- virtual ~**ByteArrayPerspective** ()
- **ByteArrayPerspective * takeRef** ()
Takes a reference to this Perspective and returns a pointer to this so that a caller can take a ref and get a ref in one call.
- void **returnRef** ()
Returns a reference to this Perspective, when the count is zero it should be deleted.
- int **getReferences** () const

6.150.1 Detailed Description

This class extends ByteArray to create a reference counted byte array that can be held and used by several different ByteBuffers and allow them to know on destruction whose job it is to delete the perspective. Creating an instance of this class implicitly takes a reference to it, so a creator must return its ref before the count will be zero.

6.150.2 Constructor & Destructor Documentation

6.150.2.1 `decaf::internal::nio::ByteArrayPerspective::ByteArrayPerspective (std::size_t capacity)`

Creates a byte array object that is allocated internally and is then owned and deleted when this object is deleted.

The array is initially created with all elements initialized to zero.

Parameters

capacity - size of the array, this is the limit we read and write to.

6.150.2.2 `decaf::internal::nio::ByteArrayPerspective::ByteArrayPerspective (unsigned char * array, std::size_t capacity, bool own = false) throw (lang::exceptions::NullPointerException)`

Creates a byte array object that wraps the given array.

If the own flag is set then it will delete this array when this object is deleted.

Parameters

array - array to wrap

capacity - size of the array, this is the limit we read and write to.

own - is this class now the owner of the pointer.

Exceptions

NullPointerException if buffer is NULL

6.150.2.3 `decaf::internal::nio::ByteArrayPerspective::ByteArrayPerspective (char * array, std::size_t capacity, bool own = false) throw (lang::exceptions::NullPointerException)`

Creates a array object that wraps the given array.

If the own flag is set then it will delete this array when this object is deleted.

Parameters

array - array to wrap

capacity - size of the array, this is the limit we read and write to.

own - is this class now the owner of the pointer.

Exceptions

NullPointerException if buffer is NULL

6.150.2.4 decaf::internal::nio::ByteArrayPerspective::ByteArrayPerspective (double * *array*, std::size_t *capacity*, bool *own* = *false*) throw (lang::exceptions::NullPointerException)

Creates a array object that wraps the given array.

If the own flag is set then it will delete this array when this object is deleted.

Parameters

array - array to wrap

capacity - size of the array, this is the limit we read and write to.

own - is this class now the owner of the pointer.

Exceptions

NullPointerException if buffer is NULL

6.150.2.5 decaf::internal::nio::ByteArrayPerspective::ByteArrayPerspective (float * *array*, std::size_t *capacity*, bool *own* = *false*) throw (lang::exceptions::NullPointerException)

Creates a array object that wraps the given array.

If the own flag is set then it will delete this array when this object is deleted.

Parameters

array - array to wrap

capacity - size of the array, this is the limit we read and write to.

own - is this class now the owner of the pointer.

Exceptions

NullPointerException if buffer is NULL

6.150.2.6 decaf::internal::nio::ByteArrayPerspective::ByteArrayPerspective (long * *array*, std::size_t *capacity*, bool *own* = *false*) throw (lang::exceptions::NullPointerException)

Creates a array object that wraps the given array.

If the own flag is set then it will delete this array when this object is deleted.

Parameters

array - array to wrap

capacity - size of the array, this is the limit we read and write to.

own - is this class now the owner of the pointer.

Exceptions

NullPointerException if buffer is NULL

6.150.2.7 `decaf::internal::nio::ByteArrayPerspective::ByteArrayPerspective (int * array, std::size_t capacity, bool own = false) throw (lang::exceptions::NullPointerException)`

Creates a array object that wraps the given array.

If the own flag is set then it will delete this array when this object is deleted.

Parameters

array - array to wrap

capacity - size of the array, this is the limit we read and write to.

own - is this class now the owner of the pointer.

Exceptions

NullPointerException if buffer is NULL

6.150.2.8 `decaf::internal::nio::ByteArrayPerspective::ByteArrayPerspective (short * array, std::size_t capacity, bool own = false) throw (lang::exceptions::NullPointerException)`

Creates a array object that wraps the given array.

If the own flag is set then it will delete this array when this object is deleted.

Parameters

array - array to wrap

capacity - size of the array, this is the limit we read and write to.

own - is this class now the owner of the pointer.

Exceptions

NullPointerException if buffer is NULL

6.150.2.9 `virtual
decaf::internal::nio::ByteArrayPerspective::~~ByteArrayPerspective ()
[inline, virtual]`

6.150.3 Member Function Documentation

6.150.3.1 `int decaf::internal::nio::ByteArrayPerspective::getReferences () const
[inline]`

Returns

the current number of reference on this perspective

6.150.3.2 `void decaf::internal::nio::ByteArrayPerspective::returnRef () [inline]`

Returns a reference to this Perspective, when the count is zero it should be deleted.

6.150.3.3 ByteArrayPerspective* decaf::internal::nio::ByteBufferPerspective::takeRef () [inline]

Takes a reference to this Perspective and returns a pointer to this so that a caller can take a ref and get a ref in one call.

Returns

this

The documentation for this class was generated from the following file:

- src/main/decaf/internal/nio/ByteBufferPerspective.h

6.151 decaf::nio::ByteBuffer Class Reference

This class defines six categories of operations upon byte buffers:

```
#include <src/main/decaf/nio/ByteBuffer.h>
```

Inheritance diagram for decaf::nio::ByteBuffer:

Public Member Functions

- virtual **~ByteBuffer** ()
- virtual std::string **toString** () const
- **ByteBuffer** & **get** (std::vector< unsigned char > buffer) throw (BufferUnderflowException)
Relative bulk get method.
- **ByteBuffer** & **get** (unsigned char *buffer, std::size_t offset, std::size_t length) throw (BufferUnderflowException, lang::exceptions::NullPointerException)
Relative bulk get method.
- **ByteBuffer** & **put** (**ByteBuffer** &src) throw (BufferOverflowException, ReadOnlyBufferException, lang::exceptions::IllegalArgumentException)
This method transfers the bytes remaining in the given source buffer into this buffer.
- **ByteBuffer** & **put** (const unsigned char *buffer, std::size_t offset, std::size_t length) throw (BufferOverflowException, ReadOnlyBufferException, lang::exceptions::NullPointerException)
This method transfers bytes into this buffer from the given source array.
- **ByteBuffer** & **put** (std::vector< unsigned char > &buffer) throw (BufferOverflowException, ReadOnlyBufferException)
This method transfers the entire content of the given source byte array into this buffer.
- virtual bool **isReadOnly** () const =0

Tells whether or not this buffer is read-only.

- virtual unsigned char * **array** ()=0 throw (ReadOnlyBufferException, lang::exceptions::UnsupportedOperationException)

Returns the byte array that backs this buffer.

- virtual std::size_t **arrayOffset** () const =0 throw (ReadOnlyBufferException, lang::exceptions::UnsupportedOperationException)

Returns the offset within this buffer's backing array of the first element of the buffer.

- virtual bool **hasArray** () const =0

Tells whether or not this buffer is backed by an accessible byte array.

- virtual **CharBuffer** * **asCharBuffer** () const =0

Creates a view of this byte buffer as a char buffer.

- virtual **DoubleBuffer** * **asDoubleBuffer** () const =0

Creates a view of this byte buffer as a double buffer.

- virtual **FloatBuffer** * **asFloatBuffer** () const =0

Creates a view of this byte buffer as a float buffer.

- virtual **IntBuffer** * **asIntBuffer** () const =0

Creates a view of this byte buffer as a int buffer.

- virtual **LongBuffer** * **asLongBuffer** () const =0

Creates a view of this byte buffer as a long buffer.

- virtual **ShortBuffer** * **asShortBuffer** () const =0

Creates a view of this byte buffer as a short buffer.

- virtual **ByteBuffer** * **asReadOnlyBuffer** () const =0

Creates a new, read-only byte buffer that shares this buffer's content.

- virtual **ByteBuffer** & **compact** ()=0 throw (ReadOnlyBufferException)

Compacts this buffer.

- virtual **ByteBuffer** * **duplicate** ()=0

Creates a new byte buffer that shares this buffer's content.

- virtual unsigned char **get** () const =0 throw (BufferUnderflowException)

Relative get method.

- virtual unsigned char **get** (std::size_t index) const =0 throw (lang::exceptions::IndexOutOfBoundsException)

Absolute get method.

- virtual char **getChar** ()=0 throw (BufferUnderflowException)

Reads the next byte at this buffer's current position, and then increments the position by one.

- virtual char **getChar** (std::size_t index) const =0 throw (lang::exceptions::IndexOutOfBoundsException)
Reads one byte at the given index and returns it.
- virtual double **getDouble** ()=0 throw (BufferUnderflowException)
Reads the next eight bytes at this buffer's current position, and then increments the position by that amount.
- virtual double **getDouble** (std::size_t index) const =0 throw (lang::exceptions::IndexOutOfBoundsException)
Reads eight bytes at the given index and returns it.
- virtual float **getFloat** ()=0 throw (BufferUnderflowException)
Reads the next four bytes at this buffer's current position, and then increments the position by that amount.
- virtual float **getFloat** (std::size_t index) const =0 throw (lang::exceptions::IndexOutOfBoundsException)
Reads four bytes at the given index and returns it.
- virtual long long **getLong** ()=0 throw (BufferUnderflowException)
Reads the next eight bytes at this buffer's current position, and then increments the position by that amount.
- virtual long long **getLong** (std::size_t index) const =0 throw (lang::exceptions::IndexOutOfBoundsException)
Reads eight bytes at the given index and returns it.
- virtual int **getInt** ()=0 throw (BufferUnderflowException)
Reads the next four bytes at this buffer's current position, and then increments the position by that amount.
- virtual int **getInt** (std::size_t index) const =0 throw (lang::exceptions::IndexOutOfBoundsException)
Reads four bytes at the given index and returns it.
- virtual short **getShort** ()=0 throw (BufferUnderflowException)
Reads the next two bytes at this buffer's current position, and then increments the position by that amount.
- virtual short **getShort** (std::size_t index) const =0 throw (lang::exceptions::IndexOutOfBoundsException)
Reads two bytes at the given index and returns it.
- virtual **ByteBuffer** & **put** (unsigned char value)=0 throw (BufferOverflowException, ReadOnlyBufferException)
Writes the given byte into this buffer at the current position, and then increments the position.
- virtual **ByteBuffer** & **put** (std::size_t index, unsigned char value)=0 throw (lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException)
Writes the given byte into this buffer at the given index.

- virtual **ByteBuffer** & **putChar** (char value)=0 throw (**BufferOverflowException**, **ReadOnlyBufferException**)
Writes one byte containing the given value, into this buffer at the current position, and then increments the position by one.
- virtual **ByteBuffer** & **putChar** (std::size_t index, char value)=0 throw (**lang::exceptions::IndexOutOfBoundsException**, **ReadOnlyBufferException**)
Writes one byte containing the given value, into this buffer at the given index.
- virtual **ByteBuffer** & **putDouble** (double value)=0 throw (**BufferOverflowException**, **ReadOnlyBufferException**)
Writes eight bytes containing the given value, into this buffer at the current position, and then increments the position by eight.
- virtual **ByteBuffer** & **putDouble** (std::size_t index, double value)=0 throw (**lang::exceptions::IndexOutOfBoundsException**, **ReadOnlyBufferException**)
Writes eight bytes containing the given value, into this buffer at the given index.
- virtual **ByteBuffer** & **putFloat** (float value)=0 throw (**BufferOverflowException**, **ReadOnlyBufferException**)
Writes four bytes containing the given value, into this buffer at the current position, and then increments the position by eight.
- virtual **ByteBuffer** & **putFloat** (std::size_t index, float value)=0 throw (**lang::exceptions::IndexOutOfBoundsException**, **ReadOnlyBufferException**)
Writes four bytes containing the given value, into this buffer at the given index.
- virtual **ByteBuffer** & **putLong** (long long value)=0 throw (**BufferOverflowException**, **ReadOnlyBufferException**)
Writes eight bytes containing the given value, into this buffer at the current position, and then increments the position by eight.
- virtual **ByteBuffer** & **putLong** (std::size_t index, long long value)=0 throw (**lang::exceptions::IndexOutOfBoundsException**, **ReadOnlyBufferException**)
Writes eight bytes containing the given value, into this buffer at the given index.
- virtual **ByteBuffer** & **putInt** (int value)=0 throw (**BufferOverflowException**, **ReadOnlyBufferException**)
Writes four bytes containing the given value, into this buffer at the current position, and then increments the position by eight.
- virtual **ByteBuffer** & **putInt** (std::size_t index, int value)=0 throw (**lang::exceptions::IndexOutOfBoundsException**, **ReadOnlyBufferException**)
Writes four bytes containing the given value, into this buffer at the given index.
- virtual **ByteBuffer** & **putShort** (short value)=0 throw (**BufferOverflowException**, **ReadOnlyBufferException**)
Writes two bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

- virtual **ByteBuffer** & **putShort** (std::size_t index, short value)=0 throw (lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException)
Writes two bytes containing the given value, into this buffer at the given index.
- virtual **ByteBuffer** * **slice** () const =0
Creates a new byte buffer whose content is a shared subsequence of this buffer's content.
- virtual int **compareTo** (const **ByteBuffer** &value) const
Compares this object with the specified object for order.
- virtual bool **equals** (const **ByteBuffer** &value) const
- virtual bool **operator==** (const **ByteBuffer** &value) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const **ByteBuffer** &value) const
Compares this object to another and returns true if this object is considered to be less than the one passed.

Static Public Member Functions

- static **ByteBuffer** * **allocate** (std::size_t capacity)
Allocates a new byte buffer whose position will be zero its limit will be its capacity and its mark is not set.
- static **ByteBuffer** * **wrap** (unsigned char *array, std::size_t offset, std::size_t length) throw (lang::exceptions::NullPointerException)
*Wraps the passed buffer with a new **ByteBuffer** (p. 848).*
- static **ByteBuffer** * **wrap** (std::vector< unsigned char > &buffer)
*Wraps the passed STL Byte Vector in a **ByteBuffer** (p. 848).*

Protected Member Functions

- **ByteBuffer** (std::size_t capacity)
*Creates a **ByteBuffer** (p. 848) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*

6.151.1 Detailed Description

This class defines six categories of operations upon byte buffers: 1. Absolute and relative get and put methods that read and write single bytes; 2. Relative bulk get methods that transfer contiguous sequences of bytes from this buffer into an array; 3. Relative bulk put methods that transfer contiguous sequences of bytes from a byte array or some other byte buffer into this buffer; 4. Absolute and relative get and put methods that read and write values of other primitive types, translating them to and from sequences of bytes in a particular byte order; 5. Methods for creating view buffers, which allow a byte buffer to be viewed as a buffer containing values of some other primitive type; and 6. Methods for compacting, duplicating, and slicing a byte buffer.

Byte buffers can be created either by allocation, which allocates space for the buffer's content, or by wrapping an existing byte array into a buffer.

Access to binary data:

This class defines methods for reading and writing values of all other primitive types, except boolean. Primitive values are translated to (or from) sequences of bytes according to the buffer's current byte order.

For access to heterogeneous binary data, that is, sequences of values of different types, this class defines a family of absolute and relative get and put methods for each type. For 32-bit floating-point values, for example, this class defines:

float **getFloat()** (p.861) float getFloat(int index) void **putFloat(float f)** (p.869) void putFloat(int index, float f)

Corresponding methods are defined for the types char, short, int, long, and double. The index parameters of the absolute get and put methods are in terms of bytes rather than of the type being read or written.

For access to homogeneous binary data, that is, sequences of values of the same type, this class defines methods that can create views of a given byte buffer. A view buffer is simply another buffer whose content is backed by the byte buffer. Changes to the byte buffer's content will be visible in the view buffer, and vice versa; the two buffers' position, limit, and mark values are independent. The asFloatBuffer method, for example, creates an instance of the **FloatBuffer** (p.1562) class that is backed by the byte buffer upon which the method is invoked. Corresponding view-creation methods are defined for the types char, short, int, long, and double.

View buffers have two important advantages over the families of type-specific get and put methods described above:

A view buffer is indexed not in terms of bytes but rather in terms of the type-specific size of its values;

A view buffer provides relative bulk get and put methods that can transfer contiguous sequences of values between a buffer and an array or some other buffer of the same type; and

6.151.2 Constructor & Destructor Documentation

6.151.2.1 decaf::nio::ByteBuffer::ByteBuffer (std::size_t *capacity*) [protected]

Creates a **ByteBuffer** (p.848) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.

The array is initially created with all elements initialized to zero.

Parameters

capacity - size of the array, this is the limit we read and write to.

6.151.2.2 virtual decaf::nio::ByteBuffer::~~ByteBuffer () [inline, virtual]

6.151.3 Member Function Documentation

6.151.3.1 static ByteBuffer* decaf::nio::ByteBuffer::allocate (std::size_t *capacity*) [static]

Allocates a new byte buffer whose position will be zero its limit will be its capacity and its mark is not set.

Parameters

capacity - the internal buffer's capacity.

Returns

a newly allocated **ByteBuffer** (p. 848) which the caller owns.

6.151.3.2 virtual unsigned char* decaf::nio::ByteBuffer::array () throw (ReadOnlyBufferException, lang::exceptions::UnsupportedOperationException) [pure virtual]

Returns the byte array that backs this buffer.

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

The array that backs this buffer

Exceptions

ReadOnlyBufferException (p. 2520) - If this buffer is backed by an array but is read-only

UnsupportedOperationException - If this buffer is not backed by an accessible array

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 812).

6.151.3.3 virtual std::size_t decaf::nio::ByteBuffer::arrayOffset () const throw (ReadOnlyBufferException, lang::exceptions::UnsupportedOperationException) [pure virtual]

Returns the offset within this buffer's backing array of the first element of the buffer.

If this buffer is backed by an array then buffer position `p` corresponds to array index `p + arrayOffset()` (p. 854).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

The offset within this buffer's array of the first element of the buffer

Exceptions

ReadOnlyBufferException (p. 2520) - If this buffer is backed by an array but is read-only

UnsupportedOperationException - If this buffer is not backed by an accessible array

Implemented in **decaf::internal::nio::ByteBuffer** (p. 812).

6.151.3.4 **virtual CharBuffer* decaf::nio::ByteBuffer::asCharBuffer () const**
[pure virtual]

Creates a view of this byte buffer as a char buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the new Char **Buffer** (p. 741), which the caller then owns.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 813).

6.151.3.5 **virtual DoubleBuffer* decaf::nio::ByteBuffer::asDoubleBuffer () const**
[pure virtual]

Creates a view of this byte buffer as a double buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by eight, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the new double **Buffer** (p. 741), which the caller then owns.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 813).

6.151.3.6 **virtual FloatBuffer* decaf::nio::ByteBuffer::asFloatBuffer () const**
[pure virtual]

Creates a view of this byte buffer as a float buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by four, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the new float **Buffer** (p. 741), which the caller then owns.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 813).

6.151.3.7 virtual IntBuffer* decaf::nio::ByteBuffer::asIntBuffer () const [pure virtual]

Creates a view of this byte buffer as a int buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by four, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the new int **Buffer** (p. 741), which the caller then owns.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 814).

6.151.3.8 virtual LongBuffer* decaf::nio::ByteBuffer::asLongBuffer () const [pure virtual]

Creates a view of this byte buffer as a long buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by eight, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the new long **Buffer** (p. 741), which the caller then owns.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 814).

6.151.3.9 virtual ByteBuffer* decaf::nio::ByteBuffer::asReadOnlyBuffer () const [pure virtual]

Creates a new, read-only byte buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns

The new, read-only byte buffer which the caller then owns.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 814).

6.151.3.10 **virtual ShortBuffer* decaf::nio::ByteBuffer::asShortBuffer () const**
[pure virtual]

Creates a view of this byte buffer as a short buffer.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by two, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the new short **Buffer** (p. 741), which the caller then owns.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 815).

6.151.3.11 **virtual ByteBuffer& decaf::nio::ByteBuffer::compact () throw (**
ReadOnlyBufferException) [pure virtual]

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index $p = \text{position}()$ (p. 746) is copied to index zero, the byte at index $p + 1$ is copied to index one, and so forth until the byte at index $\text{limit}()$ (p. 745) - 1 is copied to index $n = \text{limit}()$ (p. 745) - 1 - p . The buffer's position is then set to $n+1$ and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns

a reference to this **ByteBuffer** (p. 848)

Exceptions

ReadOnlyBufferException (p. 2520) - If this buffer is read-only

Implemented in **decaf::internal::nio::ByteBuffer** (p. 815).

6.151.3.12 **virtual int decaf::nio::ByteBuffer::compareTo (const ByteBuffer &**
value) const [virtual]

Compares this object with the specified object for order.

Returns a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

Parameters

value - the Object to be compared.

Returns

a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

6.151.3.13 virtual ByteBuffer* decaf::nio::ByteBuffer::duplicate () [pure virtual]

Creates a new byte buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

a new Byte **Buffer** (p. 741) which the caller owns.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 816).

6.151.3.14 virtual bool decaf::nio::ByteBuffer::equals (const ByteBuffer & value) const [virtual]

Returns

true if this value is considered equal to the passed value.

6.151.3.15 ByteBuffer& decaf::nio::ByteBuffer::get (std::vector< unsigned char > buffer) throw (BufferUnderflowException)

Relative bulk get method.

This method transfers bytes from this buffer into the given destination vector. An invocation of this method of the form `src.get(a)` behaves in exactly the same way as the invocation. The vector must be sized to the amount of data that is to be read, that is to say, the caller should call `buffer.resize(N)` before calling this get method.

Returns

a reference to this Byte **Buffer** (p. 741)

Exceptions

BufferUnderflowException (p. 774) - If there are fewer than length bytes remaining in this buffer

6.151.3.16 `virtual unsigned char decaf::nio::ByteBuffer::get () const throw (BufferUnderflowException)` [pure virtual]

Relative get method.

Reads the byte at this buffer's current position, and then increments the position.

Returns

The byte at the buffer's current position

Exceptions

BufferUnderflowException (p. 774) - If the buffer's current position is not smaller than its limit

Implemented in `decaf::internal::nio::ByteBuffer` (p. 816).

6.151.3.17 `virtual unsigned char decaf::nio::ByteBuffer::get (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)` [pure virtual]

Absolute get method.

Reads the byte at the given index.

Parameters

index - the index in the **Buffer** (p. 741) where the byte is to be read

Returns

the byte that is located at the given index

Exceptions

IndexOutOfBoundsException - If index is not smaller than the buffer's limit

Implemented in `decaf::internal::nio::ByteBuffer` (p. 816).

6.151.3.18 `ByteBuffer& decaf::nio::ByteBuffer::get (unsigned char * buffer, std::size_t offset, std::size_t length) throw (BufferUnderflowException, lang::exceptions::NullPointerException)`

Relative bulk get method.

This method transfers bytes from this buffer into the given destination array. If there are fewer bytes remaining in the buffer than are required to satisfy the request, that is, if `length > remaining()` (p. 746), then no bytes are transferred and a **BufferUnderflowException** (p. 774) is thrown.

Otherwise, this method copies `length` bytes from this buffer into the given array, starting at the current position of this buffer and at the given offset in the array. The position of this buffer is then incremented by `length`.

Parameters

buffer - pointer to an allocated buffer to fill

offset - position in the buffer to start filling

length - amount of data to put in the passed buffer

Returns

a reference to this **Buffer** (p. 741)

Exceptions

BufferUnderflowException (p. 774) - If there are fewer than length bytes remaining in this buffer

NullPointerException if the passed buffer is null.

6.151.3.19 virtual char decaf::nio::ByteBuffer::getChar () throw (BufferUnderflowException) [pure virtual]

Reads the next byte at this buffer's current position, and then increments the position by one.

Returns

the next char in the buffer..

Exceptions

BufferUnderflowException (p. 774) - If there are no more bytes remaining in this buffer, meaning we have reached the set limit.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 817).

6.151.3.20 virtual char decaf::nio::ByteBuffer::getChar (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException) [pure virtual]

Reads one byte at the given index and returns it.

Parameters

index - the index in the **Buffer** (p. 741) where the byte is to be read

Returns

the char at the given index in the buffer

Exceptions

IndexOutOfBoundsException - If index is not smaller than the buffer's limit

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 817).

6.151.3.21 `virtual double decaf::nio::ByteBuffer::getDouble () throw (BufferUnderflowException)` [pure virtual]

Reads the next eight bytes at this buffer's current position, and then increments the position by that amount.

Returns

the next double in the buffer..

Exceptions

BufferUnderflowException (p. 774) - If there are no more bytes remaining in this buffer, meaning we have reached the set limit.

Implemented in `decaf::internal::nio::ByteBuffer` (p. 817).

6.151.3.22 `virtual double decaf::nio::ByteBuffer::getDouble (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)` [pure virtual]

Reads eight bytes at the given index and returns it.

Parameters

index - the index in the `Buffer` (p. 741) where the bytes are to be read

Returns

the double at the given index in the buffer

Exceptions

IndexOutOfBoundsException - If there are not enough bytes remaining to fill the requested Data Type

Implemented in `decaf::internal::nio::ByteBuffer` (p. 818).

6.151.3.23 `virtual float decaf::nio::ByteBuffer::getFloat () throw (BufferUnderflowException)` [pure virtual]

Reads the next four bytes at this buffer's current position, and then increments the position by that amount.

Returns

the next float in the buffer..

Exceptions

BufferUnderflowException (p. 774) - If there are no more bytes remaining in this buffer, meaning we have reached the set limit.

Implemented in `decaf::internal::nio::ByteBuffer` (p. 818).

6.151.3.24 `virtual float decaf::nio::ByteBuffer::getFloat (std::size_t index)
const throw (lang::exceptions::IndexOutOfBoundsException) [pure
virtual]`

Reads four bytes at the given index and returns it.

Parameters

index - the index in the **Buffer** (p. 741) where the bytes are to be read

Returns

the float at the given index in the buffer

Exceptions

IndexOutOfBoundsException - If there are not enough bytes remaining to fill the requested Data Type

Implemented in **decaf::internal::nio::ByteBuffer** (p. 818).

6.151.3.25 `virtual int decaf::nio::ByteBuffer::getInt () throw (
BufferUnderflowException) [pure virtual]`

Reads the next four bytes at this buffer's current position, and then increments the position by that amount.

Returns

the next int in the buffer..

Exceptions

BufferUnderflowException (p. 774) - If there are no more bytes remaining in this buffer, meaning we have reached the set limit.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 819).

6.151.3.26 `virtual int decaf::nio::ByteBuffer::getInt (std::size_t index)
const throw (lang::exceptions::IndexOutOfBoundsException) [pure
virtual]`

Reads four bytes at the given index and returns it.

Parameters

index - the index in the **Buffer** (p. 741) where the bytes are to be read

Returns

the int at the given index in the buffer

Exceptions

IndexOutOfBoundsException - If there are not enough bytes remaining to fill the requested Data Type

Implemented in **decaf::internal::nio::ByteBuffer** (p. 819).

6.151.3.27 `virtual long long decaf::nio::ByteBuffer::getLong () throw (BufferUnderflowException) [pure virtual]`

Reads the next eight bytes at this buffer's current position, and then increments the position by that amount.

Returns

the next long long in the buffer..

Exceptions

BufferUnderflowException (p. 774) - If there are no more bytes remaining in this buffer, meaning we have reached the set limit.

Implemented in `decaf::internal::nio::ByteArrayBuffer` (p. 819).

6.151.3.28 `virtual long long decaf::nio::ByteBuffer::getLong (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException) [pure virtual]`

Reads eight bytes at the given index and returns it.

Parameters

index - the index in the **Buffer** (p. 741) where the bytes are to be read

Returns

the long long at the given index in the buffer

Exceptions

IndexOutOfBoundsException - If there are not enough bytes remaining to fill the requested Data Type

Implemented in `decaf::internal::nio::ByteArrayBuffer` (p. 820).

6.151.3.29 `virtual short decaf::nio::ByteBuffer::getShort () throw (BufferUnderflowException) [pure virtual]`

Reads the next two bytes at this buffer's current position, and then increments the position by that amount.

Returns

the next short in the buffer..

Exceptions

BufferUnderflowException (p. 774) - If there are no more bytes remaining in this buffer, meaning we have reached the set limit.

Implemented in `decaf::internal::nio::ByteArrayBuffer` (p. 820).

6.151.3.30 `virtual short decaf::nio::ByteBuffer::getShort (std::size_t index)
 const throw (lang::exceptions::IndexOutOfBoundsException) [pure
 virtual]`

Reads two bytes at the given index and returns it.

Parameters

index - the index in the **Buffer** (p. 741) where the bytes are to be read

Returns

the short at the given index in the buffer

Exceptions

IndexOutOfBoundsException - If there are not enough bytes remaining to fill the requested Data Type

Implemented in **decaf::internal::nio::ByteBuffer** (p. 820).

6.151.3.31 `virtual bool decaf::nio::ByteBuffer::hasArray () const [pure virtual]`

Tells whether or not this buffer is backed by an accessible byte array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Sub-classes should override this method if they do not have a backing array as this class always returns true.

Returns

true if, and only if, this buffer is backed by an array and is not read-only

Implemented in **decaf::internal::nio::ByteBuffer** (p. 821).

6.151.3.32 `virtual bool decaf::nio::ByteBuffer::isReadOnly () const [pure
 virtual]`

Tells whether or not this buffer is read-only.

Returns

true if, and only if, this buffer is read-only

Implemented in **decaf::internal::nio::ByteBuffer** (p. 821).

6.151.3.33 `virtual bool decaf::nio::ByteBuffer::operator< (const ByteBuffer &
 value) const [virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

Parameters

value - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

6.151.3.34 `virtual bool decaf::nio::ByteBuffer::operator==(const ByteBuffer & value) const` [virtual]

Compares equality between this object and the one passed.

Parameters

value - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

6.151.3.35 `virtual ByteBuffer& decaf::nio::ByteBuffer::put (unsigned char value) throw (BufferOverflowException, ReadOnlyBufferException)` [pure virtual]

Writes the given byte into this buffer at the current position, and then increments the position.

Parameters

value - the byte value to be written

Returns

a reference to this buffer

Exceptions

BufferOverflowException (p. 771) - If this buffer's current position is not smaller than its limit

ReadOnlyBufferException (p. 2520) - If this buffer is read-only

Implemented in `decaf::internal::nio::ByteArrayBuffer` (p. 821).

6.151.3.36 `virtual ByteBuffer& decaf::nio::ByteBuffer::put (std::size_t index, unsigned char value) throw (lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException)` [pure virtual]

Writes the given byte into this buffer at the given index.

Parameters

index - position in the **Buffer** (p. 741) to write the data

value - the byte to write.

Returns

a reference to this buffer

Exceptions

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException (p. 2520) - If this buffer is read-only

Implemented in **decaf::internal::nio::ByteBuffer** (p. 821).

6.151.3.37 ByteBuffer& decaf::nio::ByteBuffer::put (ByteBuffer & src) throw (BufferOverflowException, ReadOnlyBufferException, lang::exceptions::IllegalArgumentException)

This method transfers the bytes remaining in the given source buffer into this buffer.

If there are more bytes remaining in the source buffer than in this buffer, that is, if `src.remaining() > remaining()` (p. 746), then no bytes are transferred and a **BufferOverflowException** (p. 771) is thrown.

Otherwise, this method copies `n = src.remaining()` bytes from the given buffer into this buffer, starting at each buffer's current position. The positions of both buffers are then incremented by `n`.

Parameters

src - the buffer to take bytes from an place in this one.

Returns

a reference to this buffer

Exceptions

BufferOverflowException (p. 771) - If there is insufficient space in this buffer for the remaining bytes in the source buffer

IllegalArgumentException - If the source buffer is this buffer

ReadOnlyBufferException (p. 2520) - If this buffer is read-only

6.151.3.38 ByteBuffer& decaf::nio::ByteBuffer::put (const unsigned char * buffer, std::size_t offset, std::size_t length) throw (BufferOverflowException, ReadOnlyBufferException, lang::exceptions::NullPointerException)

This method transfers bytes into this buffer from the given source array.

If there are more bytes to be copied from the array than remain in this buffer, that is, if `length > remaining()` (p. 746), then no bytes are transferred and a **BufferOverflowException** (p. 771) is thrown.

Otherwise, this method copies `length` bytes from the given array into this buffer, starting at the given offset in the array and at the current position of this buffer. The position of this buffer is then incremented by `length`.

Parameters

buffer - The array from which bytes are to be read
offset - The offset within the array of the first byte to be read;
length - The number of bytes to be read from the given array

Returns

a reference to this buffer

Exceptions

BufferOverflowException (p. 771) - If there is insufficient space in this buffer
ReadOnlyBufferException (p. 2520) - If this buffer is read-only
NullPointerException if the passed buffer is null.

6.151.3.39 `ByteBuffer& decaf::nio::ByteBuffer::put (std::vector< unsigned char > & buffer) throw (BufferOverflowException, ReadOnlyBufferException)`

This method transfers the entire content of the given source byte array into this buffer.

This is the same as calling `put(&buffer[0], 0, buffer.size()`

Parameters

buffer - The buffer whose contents are copied to this **ByteBuffer** (p. 848)

Returns

a reference to this buffer

Exceptions

BufferOverflowException (p. 771) - If there is insufficient space in this buffer
ReadOnlyBufferException (p. 2520) - If this buffer is read-only

6.151.3.40 `virtual ByteBuffer& decaf::nio::ByteBuffer::putChar (char value) throw (BufferOverflowException, ReadOnlyBufferException) [pure virtual]`

Writes one byte containing the given value, into this buffer at the current position, and then increments the position by one.

Parameters

value - The value to be written

Returns

a reference to this buffer

Exceptions

BufferOverflowException (p. 771) - If there are fewer than bytes remaining in this buffer than the size of the data to be written

ReadOnlyBufferException (p. 2520) - If this buffer is read-only

Implemented in **decaf::internal::nio::ByteBuffer** (p. 822).

6.151.3.41 `virtual ByteBuffer& decaf::nio::ByteBuffer::putChar (std::size_t index, char value) throw (lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException) [pure virtual]`

Writes one byte containing the given value, into this buffer at the given index.

Parameters

index - position in the **Buffer** (p. 741) to write the data

value - the value to write.

Returns

a reference to this buffer

Exceptions

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException (p. 2520) - If this buffer is read-only

Implemented in **decaf::internal::nio::ByteBuffer** (p. 822).

6.151.3.42 `virtual ByteBuffer& decaf::nio::ByteBuffer::putDouble (double value) throw (BufferOverflowException, ReadOnlyBufferException) [pure virtual]`

Writes eight bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

Parameters

value - The value to be written

Returns

a reference to this buffer

Exceptions

BufferOverflowException (p. 771) - If there are fewer than bytes remaining in this buffer than the size of the data to be written

ReadOnlyBufferException (p. 2520) - If this buffer is read-only

Implemented in **decaf::internal::nio::ByteBuffer** (p. 823).

6.151.3.43 virtual ByteBuffer& decaf::nio::ByteBuffer::putDouble
 (std::size_t *index*, double *value*) throw
 (lang::exceptions::IndexOutOfBoundsException,
 ReadOnlyBufferException) [pure virtual]

Writes eight bytes containing the given value, into this buffer at the given index.

Parameters

index - position in the **Buffer** (p. 741) to write the data
value - the value to write.

Returns

a reference to this buffer

Exceptions

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.
ReadOnlyBufferException (p. 2520) - If this buffer is read-only

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 823).

6.151.3.44 virtual ByteBuffer& decaf::nio::ByteBuffer::putFloat (float *value*)
 throw (BufferOverflowException, ReadOnlyBufferException) [pure
 virtual]

Writes four bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

Parameters

value - The value to be written

Returns

a reference to this buffer

Exceptions

BufferOverflowException (p. 771) - If there are fewer than bytes remaining in this buffer than the size of the data to be written
ReadOnlyBufferException (p. 2520) - If this buffer is read-only

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 824).

6.151.3.45 virtual ByteBuffer& decaf::nio::ByteBuffer::putFloat
 (std::size_t *index*, float *value*) throw (lang::exceptions::IndexOutOfBoundsException,
 ReadOnlyBufferException) [pure virtual]

Writes four bytes containing the given value, into this buffer at the given index.

Parameters

index - position in the **Buffer** (p. 741) to write the data
value - the value to write.

Returns

a reference to this buffer

Exceptions

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException (p. 2520) - If this buffer is read-only

Implemented in **decaf::internal::nio::ByteBuffer** (p. 824).

6.151.3.46 `virtual ByteBuffer& decaf::nio::ByteBuffer::putInt (std::size_t index,
int value) throw (lang::exceptions::IndexOutOfBoundsException,
ReadOnlyBufferException) [pure virtual]`

Writes four bytes containing the given value, into this buffer at the given index.

Parameters

index - position in the **Buffer** (p. 741) to write the data
value - the value to write.

Returns

a reference to this buffer

Exceptions

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException (p. 2520) - If this buffer is read-only

Implemented in **decaf::internal::nio::ByteBuffer** (p. 824).

6.151.3.47 `virtual ByteBuffer& decaf::nio::ByteBuffer::putInt (int value) throw
(BufferOverflowException, ReadOnlyBufferException) [pure virtual]`

Writes four bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

Parameters

value - The value to be written

Returns

a reference to this buffer

Exceptions

BufferOverflowException (p. 771) - If there are fewer than bytes remaining in this buffer than the size of the data to be written

ReadOnlyBufferException (p. 2520) - If this buffer is read-only

Implemented in **decaf::internal::nio::ByteBuffer** (p. 825).

6.151.3.48 `virtual ByteBuffer& decaf::nio::ByteBuffer::putLong (long long value) throw (BufferOverflowException, ReadOnlyBufferException) [pure virtual]`

Writes eight bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

Parameters

value - The value to be written

Returns

a reference to this buffer

Exceptions

BufferOverflowException (p. 771) - If there are fewer than bytes remaining in this buffer than the size of the data to be written

ReadOnlyBufferException (p. 2520) - If this buffer is read-only

Implemented in **decaf::internal::nio::ByteBuffer** (p. 825).

6.151.3.49 `virtual ByteBuffer& decaf::nio::ByteBuffer::putLong (std::size_t index, long long value) throw (lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException) [pure virtual]`

Writes eight bytes containing the given value, into this buffer at the given index.

Parameters

index - position in the **Buffer** (p. 741) to write the data

value - the value to write.

Returns

a reference to this buffer

Exceptions

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException (p. 2520) - If this buffer is read-only

Implemented in **decaf::internal::nio::ByteBuffer** (p. 826).

6.151.3.50 `virtual ByteBuffer& decaf::nio::ByteBuffer::putShort
(std::size_t index, short value) throw
(lang::exceptions::IndexOutOfBoundsException,
ReadOnlyBufferException) [pure virtual]`

Writes two bytes containing the given value, into this buffer at the given index.

Parameters

index - position in the **Buffer** (p. 741) to write the data
value - the value to write.

Returns

a reference to this buffer

Exceptions

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException (p. 2520) - If this buffer is read-only

Implemented in **decaf::internal::nio::ByteBuffer** (p. 827).

6.151.3.51 `virtual ByteBuffer& decaf::nio::ByteBuffer::putShort (short value)
throw (BufferOverflowException, ReadOnlyBufferException) [pure
virtual]`

Writes two bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

Parameters

value - The value to be written

Returns

a reference to this buffer

Exceptions

BufferOverflowException (p. 771) - If there are fewer than bytes remaining in this buffer than the size of the data to be written

ReadOnlyBufferException (p. 2520) - If this buffer is read-only

Implemented in **decaf::internal::nio::ByteBuffer** (p. 826).

6.151.3.52 `virtual ByteBuffer* decaf::nio::ByteBuffer::slice () const [pure
virtual]`

Creates a new byte buffer whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the newly create **ByteBuffer** (p. 848) which the caller owns.

Implemented in **decaf::internal::nio::ByteBuffer** (p. 827).

6.151.3.53 `virtual std::string decaf::nio::ByteBuffer::toString () const [virtual]`

Returns

a `std::string` describing this object

6.151.3.54 `static ByteBuffer* decaf::nio::ByteBuffer::wrap (unsigned char
* array, std::size_t offset, std::size_t length) throw (
lang::exceptions::NullPointerException) [static]`

Wraps the passed buffer with a new **ByteBuffer** (p. 848).

The new buffer will be backed by the given byte array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be `array.length`, its position will be `offset`, its limit will be `offset + length`, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

array - The array that will back the new buffer

offset - The offset of the subarray to be used

length - The length of the subarray to be used

Returns

a new **ByteBuffer** (p. 848) that is backed by buffer, caller owns.

6.151.3.55 `static ByteBuffer* decaf::nio::ByteBuffer::wrap (std::vector< unsigned
char > & buffer) [static]`

Wraps the passed STL Byte Vector in a **ByteBuffer** (p. 848).

The new buffer will be backed by the given byte array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be `buffer.size()`, its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

buffer - The vector that will back the new buffer, the vector must have been sized to the desired size already by calling `vector.resize(N)`.

Returns

a new **ByteBuffer** (p. 848) that is backed by buffer, caller owns.

The documentation for this class was generated from the following file:

- src/main/decaf/nio/ByteBuffer.h

6.152 cms::BytesMessage Class Reference

A **BytesMessage** (p. 874) object is used to send a message containing a stream of unsigned bytes.

```
#include <src/main/cms/BytesMessage.h>
```

Inheritance diagram for cms::BytesMessage:

Public Member Functions

- virtual **~BytesMessage** ()
- virtual void **setBodyBytes** (const unsigned char *buffer, std::size_t numBytes)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)
sets the bytes given to the message body.
- virtual unsigned char * **getBodyBytes** () const =0 throw (cms::MessageNotReadableException, cms::CMSEException)
Gets the bytes that are contained in this message and returns them in a newly allocated array that becomes the property of the caller.
- virtual std::size_t **getBodyLength** () const =0 throw (cms::MessageNotReadableException, cms::CMSEException)
Returns the number of bytes contained in the body of this message.
- virtual void **reset** ()=0 throw (cms::MessageFormatException, cms::CMSEException)
Puts the message body in read-only mode and repositions the stream of bytes to the beginning.
- virtual bool **readBoolean** () const =0 throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException)
Reads a Boolean from the Bytes message stream.
- virtual void **writeBoolean** (bool value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)
Writes a boolean to the bytes message stream as a 1-byte value.
- virtual unsigned char **readByte** () const =0 throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException)
Reads a Byte from the Bytes message stream.
- virtual void **writeByte** (unsigned char value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)
Writes a byte to the bytes message stream as a 1-byte value.
- virtual std::size_t **readBytes** (std::vector< unsigned char > &value) const =0 throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException)

Reads a byte array from the bytes message stream.

- virtual void **writeBytes** (const std::vector< unsigned char > &value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)

Writes a byte array to the bytes message stream using the vector size as the number of bytes to write.

- virtual std::size_t **readBytes** (unsigned char *buffer, std::size_t length) const =0 throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException)

Reads a portion of the bytes message stream.

- virtual void **writeBytes** (const unsigned char *value, std::size_t offset, std::size_t length)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)

Writes a portion of a byte array to the bytes message stream.

- virtual char **readChar** () const =0 throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException)

Reads a Char from the Bytes message stream.

- virtual void **writeChar** (char value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)

Writes a char to the bytes message stream as a 1-byte value.

- virtual float **readFloat** () const =0 throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException)

Reads a 32 bit float from the Bytes message stream.

- virtual void **writeFloat** (float value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)

Writes a float to the bytes message stream as a 4 byte value.

- virtual double **readDouble** () const =0 throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException)

Reads a 64 bit double from the Bytes message stream.

- virtual void **writeDouble** (double value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)

Writes a double to the bytes message stream as a 8 byte value.

- virtual short **readShort** () const =0 throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException)

Reads a 16 bit signed short from the Bytes message stream.

- virtual void **writeShort** (short value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)

Writes a signed short to the bytes message stream as a 2 byte value.

- virtual unsigned short **readUnsignedShort** () const =0 throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException)

Reads a 16 bit unsigned short from the Bytes message stream.

- virtual void **writeUnsignedShort** (unsigned short value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)
Writes a unsigned short to the bytes message stream as a 2 byte value.
- virtual int **readInt** () const =0 throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException)
Reads a 32 bit signed integer from the Bytes message stream.
- virtual void **writeInt** (int value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)
Writes a signed int to the bytes message stream as a 4 byte value.
- virtual long long **readLong** () const =0 throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException)
Reads a 64 bit long from the Bytes message stream.
- virtual void **writeLong** (long long value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)
Writes a long long to the bytes message stream as a 8 byte value.
- virtual std::string **readString** () const =0 throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException)
Reads an ASCII String from the Bytes message stream.
- virtual void **writeString** (const std::string &value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)
Writes an ASCII String to the Bytes message stream.
- virtual std::string **readUTF** () const =0 throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException)
*Reads an UTF String from the **BytesMessage** (p. 874) stream.*
- virtual void **writeUTF** (const std::string &value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)
*Writes an UTF String to the **BytesMessage** (p. 874) stream.*
- virtual BytesMessage * **clone** () const =0
Clones this message.

6.152.1 Detailed Description

A **BytesMessage** (p. 874) object is used to send a message containing a stream of unsigned bytes. It inherits from the **Message** (p. 2036) interface and adds a bytes message body. The receiver of the message supplies the interpretation of the bytes using the methods added by the **BytesMessage** (p. 874) interface.

The **BytesMessage** (p. 874) methods are based largely on those found in **decaf.io.DataInputStream** (p. 1291) and **decaf.io.DataOutputStream** (p. 1300).

Although the CMS API allows the use of message properties with byte messages, they are typically not used, since the inclusion of properties may affect the format.

The primitive types can be written explicitly using methods for each type. Because the C++ language is more limited when dealing with primitive types the JMS equivalent generic read and write methods that take Java objects cannot be provided in the CMS API.

When the message is first created, and when `clearBody` is called, the body of the message is in write-only mode. After the first call to `reset` has been made, the message body is in read-only mode. After a message has been sent, the client that sent it can retain and modify it without affecting the message that has been sent. The same message object can be sent multiple times. When a message has been received, the provider has called `reset` so that the message body is in read-only mode for the client.

If `clearBody` is called on a message in read-only mode, the message body is cleared and the message is in write-only mode.

If a client attempts to read a message in write-only mode, a **MessageNotReadableException** (p. 2187) is thrown.

If a client attempts to write a message in read-only mode, a **MessageNotWriteableException** (p. 2188) is thrown.

Since

1.0

6.152.2 Constructor & Destructor Documentation

6.152.2.1 `virtual cms::BytesMessage::~~BytesMessage () [inline, virtual]`

6.152.3 Member Function Documentation

6.152.3.1 `virtual BytesMessage* cms::BytesMessage::clone () const [pure virtual]`

Clones this message.

Returns

a deep copy of this message.

Exceptions

CMSException (p. 960) - if an internal error occurs while cloning the **Message** (p. 2036).

Implements **cms::Message** (p. 2041).

6.152.3.2 `virtual unsigned char* cms::BytesMessage::getBodyBytes () const throw (cms::MessageNotReadableException, cms::CMSException) [pure virtual]`

Gets the bytes that are contained in this message and returns them in a newly allocated array that becomes the property of the caller.

This is a copy of the data contained in this message, changing the value contained in this array has no effect on the data contained in this message.

Returns

pointer to a byte buffer that the call owns upon completion of this method.

Exceptions

CMSException (p. 960) - If an internal error occurs.

MessageNotReadableException (p. 2187) - If the message is in Write Only Mode.

6.152.3.3 `virtual std::size_t cms::BytesMessage::getBodyLength () const throw (cms::MessageNotReadableException, cms::CMSException) [pure virtual]`

Returns the number of bytes contained in the body of this message.

Returns

number of bytes.

Exceptions

CMSException (p. 960) - If an internal error occurs.

MessageNotReadableException (p. 2187) - If the message is in Write Only Mode.

6.152.3.4 `virtual bool cms::BytesMessage::readBoolean () const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSException) [pure virtual]`

Reads a Boolean from the Bytes message stream.

Returns

boolean value from stream

Exceptions

CMSException (p. 960) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. 2140) - if unexpected end of bytes stream has been reached.

MessageNotReadableException (p. 2187) - if the message is in write-only mode.

6.152.3.5 `virtual unsigned char cms::BytesMessage::readByte () const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSException) [pure virtual]`

Reads a Byte from the Bytes message stream.

Returns

unsigned char value from stream

Exceptions

CMSException (p. 960) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. 2140) - if unexpected end of bytes stream has been reached.

MessageNotReadableException (p. 2187) - if the message is in write-only mode.

6.152.3.6 `virtual std::size_t cms::BytesMessage::readBytes (std::vector< unsigned char > & value) const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSException) [pure virtual]`

Reads a byte array from the bytes message stream.

If the length of vector value is less than the number of bytes remaining to be read from the stream, the vector should be filled. A subsequent call reads the next increment, and so on.

If the number of bytes remaining in the stream is less than the length of vector value, the bytes should be read into the vector. The return value of the total number of bytes read will be less than the length of the vector, indicating that there are no more bytes left to be read from the stream. The next read of the stream returns -1.

Parameters

value buffer to place data in

Returns

the total number of bytes read into the buffer, or -1 if there is no more data because the end of the stream has been reached

Exceptions

CMSException (p. 960) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. 2140) - if unexpected end of bytes stream has been reached.

MessageNotReadableException (p. 2187) - if the message is in write-only mode.

6.152.3.7 `virtual std::size_t cms::BytesMessage::readBytes (unsigned char * buffer, std::size_t length) const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSException) [pure virtual]`

Reads a portion of the bytes message stream.

If the length of array value is less than the number of bytes remaining to be read from the stream, the array should be filled. A subsequent call reads the next increment, and so on.

If the number of bytes remaining in the stream is less than the length of array value, the bytes should be read into the array. The return value of the total number of bytes read will be less than the length of the array, indicating that there are no more bytes left to be read from the stream. The next read of the stream returns -1.

If length is negative, or length is greater than the length of the array value, then an `IndexOutOfBoundsException` is thrown. No bytes will be read from the stream for this exception case.

Parameters

buffer the buffer into which the data is read

length the number of bytes to read; must be less than or equal to value.length

Returns

the total number of bytes read into the buffer, or -1 if there is no more data because the end of the stream has been reached

Exceptions

CMSEException (p. 960) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. 2140) - if unexpected end of bytes stream has been reached.

MessageNotReadableException (p. 2187) - if the message is in write-only mode.

6.152.3.8 virtual char cms::BytesMessage::readChar () const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException) [pure virtual]

Reads a Char from the Bytes message stream.

Returns

char value from stream

Exceptions

CMSEException (p. 960) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. 2140) - if unexpected end of bytes stream has been reached.

MessageNotReadableException (p. 2187) - if the message is in write-only mode.

6.152.3.9 virtual double cms::BytesMessage::readDouble () const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSEException) [pure virtual]

Reads a 64 bit double from the Bytes message stream.

Returns

double value from stream

Exceptions

CMSEException (p. 960) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. 2140) - if unexpected end of bytes stream has been reached.

MessageNotReadableException (p. 2187) - if the message is in write-only mode.

6.152.3.10 `virtual float cms::BytesMessage::readFloat () const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSException) [pure virtual]`

Reads a 32 bit float from the Bytes message stream.

Returns

double value from stream

Exceptions

CMSException (p. 960) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. 2140) - if unexpected end of bytes stream has been reached.

MessageNotReadableException (p. 2187) - if the message is in write-only mode.

6.152.3.11 `virtual int cms::BytesMessage::readInt () const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSException) [pure virtual]`

Reads a 32 bit signed integer from the Bytes message stream.

Returns

int value from stream

Exceptions

CMSException (p. 960) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. 2140) - if unexpected end of bytes stream has been reached.

MessageNotReadableException (p. 2187) - if the message is in write-only mode.

6.152.3.12 `virtual long long cms::BytesMessage::readLong () const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSException) [pure virtual]`

Reads a 64 bit long from the Bytes message stream.

Returns

long long value from stream

Exceptions

CMSException (p. 960) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. 2140) - if unexpected end of bytes stream has been reached.

MessageNotReadableException (p. 2187) - if the message is in write-only mode.

6.152.3.13 virtual short cms::BytesMessage::readShort () const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSException) [pure virtual]

Reads a 16 bit signed short from the Bytes message stream.

Returns

short value from stream

Exceptions

CMSException (p. 960) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. 2140) - if unexpected end of bytes stream has been reached.

MessageNotReadableException (p. 2187) - if the message is in write-only mode.

6.152.3.14 virtual std::string cms::BytesMessage::readString () const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSException) [pure virtual]

Reads an ASCII String from the Bytes message stream.

Returns

String from stream

Exceptions

CMSException (p. 960) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. 2140) - if unexpected end of bytes stream has been reached.

MessageNotReadableException (p. 2187) - if the message is in write-only mode.

6.152.3.15 virtual unsigned short cms::BytesMessage::readUnsignedShort () const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSException) [pure virtual]

Reads a 16 bit unsigned short from the Bytes message stream.

Returns

unsigned short value from stream

Exceptions

CMSException (p. 960) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. 2140) - if unexpected end of bytes stream has been reached.

MessageNotReadableException (p. 2187) - if the message is in write-only mode.

6.152.3.16 `virtual std::string cms::BytesMessage::readUTF () const throw (cms::MessageEOFException, cms::MessageNotReadableException, cms::CMSException) [pure virtual]`

Reads an UTF String from the **BytesMessage** (p. 874) stream.

Returns

String from stream

Exceptions

CMSException (p. 960) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. 2140) - if unexpected end of bytes stream has been reached.

MessageNotReadableException (p. 2187) - if the message is in write-only mode.

6.152.3.17 `virtual void cms::BytesMessage::reset () throw (cms::MessageFormatException, cms::CMSException) [pure virtual]`

Puts the message body in read-only mode and repositions the stream of bytes to the beginning.

Exceptions

CMSException (p. 960) - If the provider fails to perform the reset operation.

MessageFormatException (p. 2141) - If the **Message** (p. 2036) has an invalid format.

6.152.3.18 `virtual void cms::BytesMessage::setBodyBytes (const unsigned char * buffer, std::size_t numBytes) throw (cms::MessageNotWriteableException, cms::CMSException) [pure virtual]`

sets the bytes given to the message body.

Parameters

buffer Byte Buffer to copy

numBytes Number of bytes in Buffer to copy

Exceptions

CMSException (p. 960) - If an internal error occurs.

MessageNotWriteableException (p. 2188) - if in Read Only Mode.

6.152.3.19 `virtual void cms::BytesMessage::writeBoolean (bool value) throw (cms::MessageNotWriteableException, cms::CMSException) [pure virtual]`

Writes a boolean to the bytes message stream as a 1-byte value.

The value true is written as the value (byte)1; the value false is written as the value (byte)0.

Parameters

value boolean to write to the stream

Exceptions

CMSEException (p. 960) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. 2188) - if the message is in read-only mode.

6.152.3.20 `virtual void cms::BytesMessage::writeByte (unsigned char value)
throw (cms::MessageNotWriteableException, cms::CMSEException)
[pure virtual]`

Writes a byte to the bytes message stream as a 1-byte value.

Parameters

value byte to write to the stream

Exceptions

CMSEException (p. 960) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. 2188) - if the message is in read-only mode.

6.152.3.21 `virtual void cms::BytesMessage::writeBytes (const
std::vector< unsigned char > & value) throw (cms::MessageNotWriteableException, cms::CMSEException) [pure
virtual]`

Writes a byte array to the bytes message stream using the vector size as the number of bytes to write.

Parameters

value bytes to write to the stream

Exceptions

CMSEException (p. 960) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. 2188) - if the message is in read-only mode.

6.152.3.22 `virtual void cms::BytesMessage::writeBytes (const unsigned
char * value, std::size_t offset, std::size_t length) throw (cms::MessageNotWriteableException, cms::CMSEException) [pure
virtual]`

Writes a portion of a byte array to the bytes message stream.

size as the number of bytes to write.

Parameters

value bytes to write to the stream
offset the initial offset within the byte array
length the number of bytes to use

Exceptions

CMSException (p. 960) - if the CMS provider fails to write the message due to some internal error.
MessageNotWriteableException (p. 2188) - if the message is in read-only mode.

6.152.3.23 `virtual void cms::BytesMessage::writeChar (char value) throw (cms::MessageNotWriteableException, cms::CMSException) [pure virtual]`

Writes a char to the bytes message stream as a 1-byte value.

Parameters

value char to write to the stream

Exceptions

CMSException (p. 960) - if the CMS provider fails to write the message due to some internal error.
MessageNotWriteableException (p. 2188) - if the message is in read-only mode.

6.152.3.24 `virtual void cms::BytesMessage::writeDouble (double value) throw (cms::MessageNotWriteableException, cms::CMSException) [pure virtual]`

Writes a double to the bytes message stream as a 8 byte value.

Parameters

value double to write to the stream

Exceptions

CMSException (p. 960) - if the CMS provider fails to write the message due to some internal error.
MessageNotWriteableException (p. 2188) - if the message is in read-only mode.

6.152.3.25 `virtual void cms::BytesMessage::writeFloat (float value) throw (cms::MessageNotWriteableException, cms::CMSException) [pure virtual]`

Writes a float to the bytes message stream as a 4 byte value.

Parameters

value float to write to the stream

Exceptions

CMSEException (p. 960) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. 2188) - if the message is in read-only mode.

6.152.3.26 `virtual void cms::BytesMessage::writeInt (int value) throw (cms::MessageNotWriteableException, cms::CMSEException) [pure virtual]`

Writes a signed int to the bytes message stream as a 4 byte value.

Parameters

value signed int to write to the stream

Exceptions

CMSEException (p. 960) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. 2188) - if the message is in read-only mode.

6.152.3.27 `virtual void cms::BytesMessage::writeLong (long long value) throw (cms::MessageNotWriteableException, cms::CMSEException) [pure virtual]`

Writes a long long to the bytes message stream as a 8 byte value.

Parameters

value signed long long to write to the stream

Exceptions

CMSEException (p. 960) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. 2188) - if the message is in read-only mode.

6.152.3.28 `virtual void cms::BytesMessage::writeShort (short value) throw (cms::MessageNotWriteableException, cms::CMSEException) [pure virtual]`

Writes a signed short to the bytes message stream as a 2 byte value.

Parameters

value signed short to write to the stream

Exceptions

CMSException (p. 960) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. 2188) - if the message is in read-only mode.

6.152.3.29 `virtual void cms::BytesMessage::writeString (const std::string & value) throw (cms::MessageNotWriteableException, cms::CMSException) [pure virtual]`

Writes an ASCII String to the Bytes message stream.

Parameters

value String to write to the stream

Exceptions

CMSException (p. 960) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. 2188) - if the message is in read-only mode.

6.152.3.30 `virtual void cms::BytesMessage::writeUnsignedShort (unsigned short value) throw (cms::MessageNotWriteableException, cms::CMSException) [pure virtual]`

Writes a unsigned short to the bytes message stream as a 2 byte value.

Parameters

value unsigned short to write to the stream

Exceptions

CMSException (p. 960) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. 2188) - if the message is in read-only mode.

6.152.3.31 `virtual void cms::BytesMessage::writeUTF (const std::string & value) throw (cms::MessageNotWriteableException, cms::CMSException) [pure virtual]`

Writes an UTF String to the **BytesMessage** (p. 874) stream.

Parameters

value String to write to the stream

Exceptions

CMSException (p. 960) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. 2140) - if unexpected end of bytes stream has been reached.

MessageNotReadableException (p. 2187) - if the message is in write-only mode.

The documentation for this class was generated from the following file:

- src/main/cms/BytesMessage.h

6.153 activemq::cmsutil::CachedConsumer Class Reference

A cached message consumer contained within a pooled session.

```
#include <src/main/activemq/cmsutil/CachedConsumer.h>
```

Inheritance diagram for activemq::cmsutil::CachedConsumer:

Public Member Functions

- **CachedConsumer** (cms::MessageConsumer *consumer)
- virtual ~**CachedConsumer** ()
- virtual void **close** () throw (cms::CMSEException)
Does nothing - the real producer resource will be closed by the lifecycle manager.
- virtual cms::Message * **receive** () throw (cms::CMSEException)
Synchronously Receive a Message.
- virtual cms::Message * **receive** (int millisecs) throw (cms::CMSEException)
Synchronously Receive a Message, time out after defined interval.
- virtual cms::Message * **receiveNoWait** () throw (cms::CMSEException)
Receive a Message, does not wait if there isn't a new message to read, returns NULL if nothing read.
- virtual void **setMessageListener** (cms::MessageListener *listener) throw (cms::CMSEException)
Sets the MessageListener that this class will send notifis on.
- virtual cms::MessageListener * **getMessageListener** () const throw (cms::CMSEException)
Gets the MessageListener that this class will send mew Message notification events to.
- virtual std::string **getMessageSelector** () const throw (cms::CMSEException)
Gets this message consumer's message selector expression.

6.153.1 Detailed Description

A cached message consumer contained within a pooled session.

6.153.2 Constructor & Destructor Documentation

6.153.2.1 `activemq::cmsutil::CachedConsumer::CachedConsumer (cms::MessageConsumer * consumer)` [inline]

6.153.2.2 `virtual activemq::cmsutil::CachedConsumer::~~CachedConsumer ()` [inline, virtual]

6.153.3 Member Function Documentation

6.153.3.1 `virtual void activemq::cmsutil::CachedConsumer::close ()` throw (cms::CMSEException) [inline, virtual]

Does nothing - the real producer resource will be closed by the lifecycle manager.

Implements `cms::Closeable` (p. 952).

6.153.3.2 `virtual cms::MessageListener* activemq::cmsutil::CachedConsumer::getMessageListener ()` const throw (cms::CMSEException) [inline, virtual]

Gets the MessageListener that this class will send new Message notification events to.

Returns

The listener of messages received by this consumer

Exceptions

CMSEException - If an internal error occurs.

Implements `cms::MessageConsumer` (p. 2081).

6.153.3.3 `virtual std::string activemq::cmsutil::CachedConsumer::getMessageSelector ()` const throw (cms::CMSEException) [inline, virtual]

Gets this message consumer's message selector expression.

Returns

This Consumer's selector expression or "".

Exceptions

CMSEException - If an internal error occurs.

Implements `cms::MessageConsumer` (p. 2081).

6.153.3.4 `virtual cms::Message* activemq::cmsutil::CachedConsumer::receive ()` throw (cms::CMSEException) [inline, virtual]

Synchronously Receive a Message.

Returns

new message which the caller owns and must delete.

Exceptions

CMSEException - If an internal error occurs.

Implements **cms::MessageConsumer** (p. 2082).

6.153.3.5 `virtual cms::Message* activemq::cmsutil::CachedConsumer::receive (int
milliseconds) throw (cms::CMSEException) [inline, virtual]`

Synchronously Receive a Message, time out after defined interval.

Returns null if nothing read.

Returns

new message which the caller owns and must delete.

Exceptions

CMSEException - If an internal error occurs.

Implements **cms::MessageConsumer** (p. 2082).

6.153.3.6 `virtual cms::Message* ac-
tivemq::cmsutil::CachedConsumer::receiveNoWait ()
throw (cms::CMSEException) [inline, virtual]`

Receive a Message, does not wait if there isn't a new message to read, returns NULL if nothing read.

Returns

new message which the caller owns and must delete.

Exceptions

CMSEException - If an internal error occurs.

Implements **cms::MessageConsumer** (p. 2082).

6.153.3.7 `virtual void activemq::cmsutil::CachedConsumer::setMessageListener
(cms::MessageListener * listener) throw (cms::CMSEException)
[inline, virtual]`

Sets the MessageListener that this class will send notifs on.

Parameters

listener The listener of messages received by this consumer.

Exceptions

CMSEException - If an internal error occurs.

Implements **cms::MessageConsumer** (p. 2082).

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/**CachedConsumer.h**

6.154 activemq::cmsutil::CachedProducer Class Reference

A cached message producer contained within a pooled session.

#include <src/main/activemq/cmsutil/CachedProducer.h>

Inheritance diagram for activemq::cmsutil::CachedProducer:

Public Member Functions

- **CachedProducer** (**cms::MessageProducer** *producer)
- virtual **~CachedProducer** ()
- virtual void **close** () throw (cms::CMSEException)
Does nothing - the real producer resource will be closed by the lifecycle manager.
- virtual void **send** (**cms::Message** *message) throw (cms::CMSEException)
Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.
- virtual void **send** (**cms::Message** *message, int deliveryMode, int priority, long long timeToLive) throw (cms::CMSEException)
Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.
- virtual void **send** (const **cms::Destination** *destination, **cms::Message** *message) throw (cms::CMSEException)
Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.
- virtual void **send** (const **cms::Destination** *destination, **cms::Message** *message, int deliveryMode, int priority, long long timeToLive) throw (cms::CMSEException)
Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.
- virtual void **setDeliveryMode** (int mode) throw (cms::CMSEException)
Sets the delivery mode for this Producer.
- virtual int **getDeliveryMode** () const throw (cms::CMSEException)
Gets the delivery mode for this Producer.

- virtual void **setDisableMessageID** (bool value) throw (cms::CMSEException)
Sets if Message Ids are disabled for this Producer.
- virtual bool **getDisableMessageID** () const throw (cms::CMSEException)
Gets if Message Ids are disabled for this Producer.
- virtual void **setDisableMessageTimeStamp** (bool value) throw (cms::CMSEException)
Sets if Message Time Stamps are disabled for this Producer.
- virtual bool **getDisableMessageTimeStamp** () const throw (cms::CMSEException)
Gets if Message Time Stamps are disabled for this Producer.
- virtual void **setPriority** (int priority) throw (cms::CMSEException)
Sets the Priority that this Producers sends messages at.
- virtual int **getPriority** () const throw (cms::CMSEException)
Gets the Priority level that this producer sends messages at.
- virtual void **setTimeToLive** (long long time) throw (cms::CMSEException)
Sets the Time to Live that this Producers sends messages with.
- virtual long long **getTimeToLive** () const throw (cms::CMSEException)
Gets the Time to Live that this producer sends messages with.

6.154.1 Detailed Description

A cached message producer contained within a pooled session.

6.154.2 Constructor & Destructor Documentation

- 6.154.2.1** `activemq::cmsutil::CachedProducer::CachedProducer (cms::MessageProducer * producer) [inline]`
- 6.154.2.2** `virtual activemq::cmsutil::CachedProducer::~~CachedProducer () [inline, virtual]`

6.154.3 Member Function Documentation

- 6.154.3.1** `virtual void activemq::cmsutil::CachedProducer::close () throw (cms::CMSEException) [inline, virtual]`

Does nothing - the real producer resource will be closed by the lifecycle manager.

Implements `cms::Closeable` (p. 952).

- 6.154.3.2** `virtual int activemq::cmsutil::CachedProducer::getDeliveryMode () const throw (cms::CMSEException) [inline, virtual]`

Gets the delivery mode for this Producer.

Returns

The DeliveryMode

Exceptions

CMSEException - if an internal error occurs.

Implements **cms::MessageProducer** (p. 2191).

6.154.3.3 `virtual bool activemq::cmsutil::CachedProducer::getDisableMessageID () const throw (cms::CMSEException) [inline, virtual]`

Gets if Message Ids are disabled for this Producer.

Returns

boolean indicating enable / disable (true / false)

Exceptions

CMSEException - if an internal error occurs.

Implements **cms::MessageProducer** (p. 2191).

6.154.3.4 `virtual bool activemq::cmsutil::CachedProducer::getDisableMessageTimeStamp () const throw (cms::CMSEException) [inline, virtual]`

Gets if Message Time Stamps are disabled for this Producer.

Returns

boolean indicating enable / disable (true / false)

Exceptions

CMSEException - if an internal error occurs.

Implements **cms::MessageProducer** (p. 2192).

6.154.3.5 `virtual int activemq::cmsutil::CachedProducer::getPriority () const throw (cms::CMSEException) [inline, virtual]`

Gets the Priority level that this producer sends messages at.

Returns

int based priority level

Exceptions

CMSEException - if an internal error occurs.

Implements **cms::MessageProducer** (p. 2192).

6.154.3.6 virtual long long activemq::cmsutil::CachedProducer::getTimeToLive () const throw (cms::CMSEException) [inline, virtual]

Gets the Time to Live that this producer sends messages with.

Returns

Time to live value in milliseconds

Exceptions

CMSEException - if an internal error occurs.

Implements **cms::MessageProducer** (p. 2192).

6.154.3.7 virtual void activemq::cmsutil::CachedProducer::send (cms::Message * message) throw (cms::CMSEException) [inline, virtual]

Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.

Uses default values for deliveryMode, priority, and time to live.

Parameters

message The message to be sent.

Exceptions

CMSEException - if an internal error occurs while sending the message.

MessageFormatException - if an Invalid Message is given.

InvalidDestinationException - if a client uses this method with a MessageProducer with an invalid destination.

UnsupportedOperationException - if a client uses this method with a MessageProducer that did not specify a destination at creation time.

Implements **cms::MessageProducer** (p. 2194).

6.154.3.8 virtual void activemq::cmsutil::CachedProducer::send (const cms::Destination * destination, cms::Message * message) throw (cms::CMSEException) [inline, virtual]

Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.

Uses default values for deliveryMode, priority, and time to live.

Parameters

destination The destination on which to send the message

message the message to be sent.

Exceptions

CMSEException - if an internal error occurs while sending the message.

MessageFormatException - if an Invalid Message is given.

InvalidDestinationException - if a client uses this method with a MessageProducer with an invalid destination.

UnsupportedOperationException - if a client uses this method with a MessageProducer that did not specify a destination at creation time.

Implements **cms::MessageProducer** (p. 2194).

6.154.3.9 `virtual void activemq::cmsutil::CachedProducer::send (const cms::Destination * destination, cms::Message * message, int deliveryMode, int priority, long long timeToLive) throw (cms::CMSEException)` [inline, virtual]

Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.

Parameters

destination The destination on which to send the message

message The message to be sent.

deliveryMode The delivery mode to be used.

priority The priority for this message.

timeToLive The time to live value for this message in milliseconds.

Exceptions

CMSEException - if an internal error occurs while sending the message.

MessageFormatException - if an Invalid Message is given.

InvalidDestinationException - if a client uses this method with a MessageProducer with an invalid destination.

UnsupportedOperationException - if a client uses this method with a MessageProducer that did not specify a destination at creation time.

Implements **cms::MessageProducer** (p. 2193).

6.154.3.10 `virtual void activemq::cmsutil::CachedProducer::send (cms::Message * message, int deliveryMode, int priority, long long timeToLive) throw (cms::CMSEException)` [inline, virtual]

Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.

Parameters

message The message to be sent.

deliveryMode The delivery mode to be used.

priority The priority for this message.

timeToLive The time to live value for this message in milliseconds.

Exceptions

CMSEException - if an internal error occurs while sending the message.

MessageFormatException - if an Invalid Message is given.

InvalidDestinationException - if a client uses this method with a MessageProducer with an invalid destination.

UnsupportedOperationException - if a client uses this method with a MessageProducer that did not specify a destination at creation time.

Implements **cms::MessageProducer** (p. 2192).

6.154.3.11 `virtual void activemq::cmsutil::CachedProducer::setDeliveryMode (int mode) throw (cms::CMSEException) [inline, virtual]`

Sets the delivery mode for this Producer.

Parameters

mode The DeliveryMode

Exceptions

CMSEException - if an internal error occurs.

Implements **cms::MessageProducer** (p. 2195).

6.154.3.12 `virtual void activemq::cmsutil::CachedProducer::setDisableMessageID (bool value) throw (cms::CMSEException) [inline, virtual]`

Sets if Message Ids are disabled for this Producer.

Parameters

value boolean indicating enable / disable (true / false)

Exceptions

CMSEException - if an internal error occurs.

Implements **cms::MessageProducer** (p. 2195).

6.154.3.13 `virtual void activemq::cmsutil::CachedProducer::setDisableMessageTimeStamp (bool value) throw (cms::CMSEException) [inline, virtual]`

Sets if Message Time Stamps are disabled for this Producer.

Parameters

value - boolean indicating enable / disable (true / false)

Exceptions

CMSEException - if an internal error occurs.

Implements **cms::MessageProducer** (p. 2195).

6.154.3.14 `virtual void activemq::cmsutil::CachedProducer::setPriority (int priority) throw (cms::CMSEException) [inline, virtual]`

Sets the Priority that this Producers sends messages at.

Parameters

priority int value for Priority level

Exceptions

CMSEException - if an internal error occurs.

Implements `cms::MessageProducer` (p. 2195).

6.154.3.15 `virtual void activemq::cmsutil::CachedProducer::setTimeToLive (long long time) throw (cms::CMSEException) [inline, virtual]`

Sets the Time to Live that this Producers sends messages with.

This value will be used if the time to live is not specified via the send method.

Parameters

time default time to live value in milliseconds

Exceptions

CMSEException - if an internal error occurs.

Implements `cms::MessageProducer` (p. 2196).

The documentation for this class was generated from the following file:

- `src/main/activemq/cmsutil/CachedProducer.h`

6.155 decaf::util::concurrent::Callable< V > Class Template Reference

A task that returns a result and may throw an exception.

```
#include <src/main/decaf/util/concurrent/Callable.h>
```

Public Member Functions

- `virtual ~Callable ()`
- `virtual V call ()=0 throw (decaf::lang::Exception)`

Computes a result, or throws an exception if unable to do so.

6.155.1 Detailed Description

`template<typename V> class decaf::util::concurrent::Callable< V >`

A task that returns a result and may throw an exception. Implementors define a single method with no arguments called `call`. This interface differs from the `Runnable` interface in that a **Callable** (p. 897) object can return a result and is allowed to throw an exceptions from its `call` method.

The `Executors` class contains utility methods to convert from other common forms to **Callable** (p. 897) classes.

Since

1.0

6.155.2 Constructor & Destructor Documentation

6.155.2.1 `template<typename V > virtual decaf::util::concurrent::Callable< V >::~~Callable ()` [inline, virtual]

6.155.3 Member Function Documentation

6.155.3.1 `template<typename V > virtual V decaf::util::concurrent::Callable< V >::call ()` throw (`decaf::lang::Exception`) [pure virtual]

Computes a result, or throws an exception if unable to do so.

Returns

Computed Result.

Exceptions

Exception If unable to compute a result.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/Callable.h`

6.156 decaf::util::concurrent::CancellationException Class Reference

```
#include <src/main/decaf/util/concurrent/CancellationException.h>
```

Inheritance diagram for `decaf::util::concurrent::CancellationException`:

Public Member Functions

- **CancellationException** () throw ()
Default Constructor.

- **CancellationException** (const decaf::lang::Exception &ex) throw ()
Conversion Constructor from some other Exception.
- **CancellationException** (const CancellationException &ex) throw ()
Copy Constructor.
- **CancellationException** (const std::exception *cause) throw ()
Constructor.
- **CancellationException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **CancellationException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **CancellationException * clone** () const
Clones this exception.
- virtual **~CancellationException** () throw ()

6.156.1 Constructor & Destructor Documentation

6.156.1.1 decaf::util::concurrent::CancellationException::CancellationException () throw () [inline]

Default Constructor.

6.156.1.2 decaf::util::concurrent::CancellationException::CancellationException (const decaf::lang::Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

Parameters

ex An exception that should become this type of Exception

References decaf::lang::Exception::Exception().

6.156.1.3 decaf::util::concurrent::CancellationException::CancellationException (const CancellationException & ex) throw () [inline]

Copy Constructor.

Parameters

ex - The Exception to copy in this new instance.

References decaf::lang::Exception::Exception().

6.156.1.4 decaf::util::concurrent::CancellationException::CancellationException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.156.1.5 decaf::util::concurrent::CancellationException::CancellationException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file - The file name where exception occurs
lineNumber - The line number where the exception occurred.
msg - The message to report
... - list of primitives that are formatted into the message

6.156.1.6 decaf::util::concurrent::CancellationException::CancellationException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file - The file name where exception occurs
lineNumber - The line number where the exception occurred.
cause - The exception that was the cause for this one to be thrown.
msg - The message to report
... - list of primitives that are formatted into the message

6.156.1.7 virtual decaf::util::concurrent::CancellationException::~~CancellationException () throw () [inline, virtual]

6.156.2 Member Function Documentation

6.156.2.1 virtual CancellationException* decaf::util::concurrent::CancellationException::clone () const [inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

a new instance of an exception that is a clone of this one.

Reimplemented from **decaf::lang::Exception** (p. 1479).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/CancellationException.h`

6.157 decaf::security::cert::Certificate Class Reference

Base interface for all identity certificates.

```
#include <src/main/decaf/security/cert/Certificate.h>
```

Inheritance diagram for `decaf::security::cert::Certificate`:

Public Member Functions

- virtual `~Certificate ()`
- virtual `bool equals (const Certificate &cert) const =0`
Compares the encoded form of the two certificates.
- virtual `void getEncoded (std::vector< unsigned char > &output) const =0 throw (CertificateEncodingException)`
Provides the encoded form of this certificate.
- virtual `std::string getType () const =0`
Returns the type of this certificate.
- virtual `PublicKey * getPublicKey ()=0`
Gets the public key of this certificate.
- virtual `const PublicKey * getPublicKey () const =0`
Gets the public key of this certificate.
- virtual `void verify (const PublicKey &publicKey) const =0 throw (NoSuchAlgorithmException, InvalidKeyException, NoSuchProviderException, SignatureException, CertificateException)`
Verifies that this certificate was signed with the private key that corresponds to the specified public key.
- virtual `void verify (const PublicKey &publicKey, const std::string &sigProvider) const =0 throw (NoSuchAlgorithmException, InvalidKeyException, NoSuchProviderException, SignatureException, CertificateException)`

Verifies that this certificate was signed with the private key that corresponds to the specified public key.

- virtual std::string **toString** () const =0
Returns a string representation of this certificate.

6.157.1 Detailed Description

Base interface for all identity certificates.

6.157.2 Constructor & Destructor Documentation

- 6.157.2.1** virtual decaf::security::cert::Certificate::~Certificate () [inline, virtual]

6.157.3 Member Function Documentation

- 6.157.3.1** virtual bool decaf::security::cert::Certificate::equals (const Certificate & *cert*) const [pure virtual]

Compares the encoded form of the two certificates.

Parameters

cert The certificate to be tested for equality with this certificate.

Returns

true if the given certificate is equal to this certificate.

- 6.157.3.2** virtual void decaf::security::cert::Certificate::getEncoded (std::vector< unsigned char > & *output*) const throw (CertificateEncodingException) [pure virtual]

Provides the encoded form of this certificate.

Parameters

output Receives the encoded form of this certificate.

Exceptions

CertificateEncodingException (p. 904) if an encoding error occurs

Implemented in `decaf::security_provider::unix::openssl::OpenSSLX509Certificate` (p. 2292).

- 6.157.3.3** virtual PublicKey* decaf::security::cert::Certificate::getPublicKey () [pure virtual]

Gets the public key of this certificate.

Returns

the public key

Implemented in `decaf::security_provider::unix::openssl::OpenSSLX509Certificate` (p. 2293).

6.157.3.4 `virtual const PublicKey* decaf::security::cert::Certificate::getPublicKey () const` [pure virtual]

Gets the public key of this certificate.

Returns

the public key

Implemented in `decaf::security_provider::unix::openssl::OpenSSLX509Certificate` (p. 2293).

6.157.3.5 `virtual std::string decaf::security::cert::Certificate::getType () const` [pure virtual]

Returns the type of this certificate.

Returns

the type of this certificate

Implemented in `decaf::security_provider::unix::openssl::OpenSSLX509Certificate` (p. 2294).

6.157.3.6 `virtual std::string decaf::security::cert::Certificate::toString () const` [pure virtual]

Returns a string representation of this certificate.

Returns

a string representation of this certificate

Implemented in `decaf::security_provider::unix::openssl::OpenSSLX509Certificate` (p. 2295).

6.157.3.7 `virtual void decaf::security::cert::Certificate::verify (const PublicKey & publicKey, const std::string & sigProvider) const throw (NoSuchAlgorithmException, InvalidKeyException, NoSuchProviderException, SignatureException, CertificateException)` [pure virtual]

Verifies that this certificate was signed with the private key that corresponds to the specified public key.

Uses the verification engine of the specified provider.

Parameters

publicKey The public key used to carry out the validation.

sigProvider The name of the signature provider

Exceptions

NoSuchAlgorithmException (p. 2272) - on unsupported signature algorithms.

InvalidKeyException (p. 1698) - on incorrect key.

NoSuchProviderException (p. 2277) - if there's no default provider.

SignatureException (p. 2779) - on signature errors.

CertificateException (p. 906) - on encoding errors.

6.157.3.8 virtual void decaf::security::cert::Certificate::verify (const PublicKey & publicKey) const throw (NoSuchAlgorithmException, InvalidKeyException, NoSuchProviderException, SignatureException, CertificateException) [pure virtual]

Verifies that this certificate was signed with the private key that corresponds to the specified public key.

Parameters

publicKey The public key used to carry out the validation.

Exceptions

NoSuchAlgorithmException (p. 2272) - on unsupported signature algorithms.

InvalidKeyException (p. 1698) - on incorrect key.

NoSuchProviderException (p. 2277) - if there's no default provider.

SignatureException (p. 2779) - on signature errors.

CertificateException (p. 906) - on encoding errors.

The documentation for this class was generated from the following file:

- src/main/decaf/security/cert/Certificate.h

6.158 decaf::security::cert::CertificateEncodingException Class Reference

```
#include <src/main/decaf/security/cert/CertificateEncodingException.h>
```

Inheritance diagram for decaf::security::cert::CertificateEncodingException:

Public Member Functions

- **CertificateEncodingException** () throw ()
Default Constructor.
- **CertificateEncodingException** (const Exception &ex) throw ()
Conversion Constructor from some other Exception.
- **CertificateEncodingException** (const **CertificateEncodingException** &ex) throw ()
Copy Constructor.
- **CertificateEncodingException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **CertificateEncodingException** * clone () const
Clones this exception.
- virtual ~**CertificateEncodingException** () throw ()

6.158.1 Constructor & Destructor Documentation

6.158.1.1 `decaf::security::cert::CertificateEncodingException::CertificateEncodingException () throw () [inline]`

Default Constructor.

6.158.1.2 `decaf::security::cert::CertificateEncodingException::CertificateEncodingException (const Exception & ex) throw () [inline]`

Conversion Constructor from some other Exception.

Parameters

ex An exception that should become this type of Exception

6.158.1.3 `decaf::security::cert::CertificateEncodingException::CertificateEncodingException (const CertificateEncodingException & ex) throw () [inline]`

Copy Constructor.

Parameters

ex An exception that should become this type of Exception

6.158.1.4 `decaf::security::cert::CertificateEncodingException::CertificateEncodingException`
 (`const char * file`, `const int lineNumber`, `const char * msg`, ...)
 throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file name where exception occurs

lineNumber line number where the exception occurred.

msg message to report

... list of primitives that are formatted into the message

6.158.1.5 `virtual`
`decaf::security::cert::CertificateEncodingException::~~CertificateEncodingException`
 () throw () [inline, virtual]

6.158.2 Member Function Documentation

6.158.2.1 `virtual CertificateEncodingException* de-`
`caf::security::cert::CertificateEncodingException::clone (`
`) const` [inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

A deep copy of this exception.

Reimplemented from `decaf::security::cert::CertificateException` (p.908).

The documentation for this class was generated from the following file:

- `src/main/decaf/security/cert/CertificateEncodingException.h`

6.159 decaf::security::cert::CertificateException Class Reference

```
#include <src/main/decaf/security/cert/CertificateException.h>
```

Inheritance diagram for `decaf::security::cert::CertificateException`:

Public Member Functions

- **CertificateException** () throw ()
Default Constructor.
- **CertificateException** (const Exception &ex) throw ()
Conversion Constructor from some other Exception.
- **CertificateException** (const **CertificateException** &ex) throw ()
Copy Constructor.
- **CertificateException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **CertificateException** * **clone** () const
Clones this exception.
- virtual ~**CertificateException** () throw ()

6.159.1 Constructor & Destructor Documentation

6.159.1.1 **decaf::security::cert::CertificateException::CertificateException ()**
throw () [inline]

Default Constructor.

6.159.1.2 **decaf::security::cert::CertificateException::CertificateException (const**
Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

Parameters

ex An exception that should become this type of Exception

6.159.1.3 **decaf::security::cert::CertificateException::CertificateException (const**
CertificateException & ex) throw () [inline]

Copy Constructor.

Parameters

ex An exception that should become this type of Exception

6.159.1.4 `decaf::security::cert::CertificateException::CertificateException (const char * file, const int lineNumber, const char * msg, ...) throw ()` [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file name where exception occurs

lineNumber line number where the exception occurred.

msg message to report

... list of primitives that are formatted into the message

6.159.1.5 `virtual decaf::security::cert::CertificateException::~~CertificateException () throw ()` [inline, virtual]

6.159.2 Member Function Documentation

6.159.2.1 `virtual CertificateException* decaf::security::cert::CertificateException::clone () const` [inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

A deep copy of this exception.

Reimplemented from `decaf::security::GeneralSecurityException` (p. 1604).

Reimplemented in `decaf::security::cert::CertificateEncodingException` (p. 906), `decaf::security::cert::CertificateExpiredException` (p. 910), `decaf::security::cert::CertificateNotYetValidException` (p. 912), and `decaf::security::cert::CertificateParsingException` (p. 914).

The documentation for this class was generated from the following file:

- `src/main/decaf/security/cert/CertificateException.h`

6.160 decaf::security::cert::CertificateExpiredException Class Reference

```
#include <src/main/decaf/security/cert/CertificateExpiredException.h>
```

Inheritance diagram for `decaf::security::cert::CertificateExpiredException`:

Public Member Functions

- **CertificateExpiredException** () throw ()
Default Constructor.
- **CertificateExpiredException** (const Exception &ex) throw ()
Conversion Constructor from some other Exception.
- **CertificateExpiredException** (const **CertificateExpiredException** &ex) throw ()
Copy Constructor.
- **CertificateExpiredException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **CertificateExpiredException** * clone () const
Clones this exception.
- virtual ~**CertificateExpiredException** () throw ()

6.160.1 Constructor & Destructor Documentation

6.160.1.1 `decaf::security::cert::CertificateExpiredException::CertificateExpiredException () throw () [inline]`

Default Constructor.

6.160.1.2 `decaf::security::cert::CertificateExpiredException::CertificateExpiredException (const Exception & ex) throw () [inline]`

Conversion Constructor from some other Exception.

Parameters

ex An exception that should become this type of Exception

6.160.1.3 `decaf::security::cert::CertificateExpiredException::CertificateExpiredException (const CertificateExpiredException & ex) throw () [inline]`

Copy Constructor.

Parameters

ex An exception that should become this type of Exception

6.160.1.4 `decaf::security::cert::CertificateExpiredException::CertificateExpiredException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file name where exception occurs

lineNumber line number where the exception occurred.

msg message to report

... list of primitives that are formatted into the message

6.160.1.5 `virtual decaf::security::cert::CertificateExpiredException::~CertificateExpiredException () throw () [inline, virtual]`

6.160.2 Member Function Documentation

6.160.2.1 `virtual CertificateExpiredException* decaf::security::cert::CertificateExpiredException::clone () const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

A deep copy of this exception.

Reimplemented from `decaf::security::cert::CertificateException` (p.908).

The documentation for this class was generated from the following file:

- `src/main/decaf/security/cert/CertificateExpiredException.h`

6.161 decaf::security::cert::CertificateNotYetValidException Class Reference

```
#include <src/main/decaf/security/cert/CertificateNotYetValidException.h>
```

Inheritance diagram for `decaf::security::cert::CertificateNotYetValidException`:

Public Member Functions

- **CertificateNotYetValidException** () throw ()
Default Constructor.
- **CertificateNotYetValidException** (const Exception &ex) throw ()
Conversion Constructor from some other Exception.
- **CertificateNotYetValidException** (const **CertificateNotYetValidException** &ex) throw ()
Copy Constructor.
- **CertificateNotYetValidException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **CertificateNotYetValidException** * clone () const
Clones this exception.
- virtual ~**CertificateNotYetValidException** () throw ()

6.161.1 Constructor & Destructor Documentation

6.161.1.1 `decaf::security::cert::CertificateNotYetValidException::CertificateNotYetValidException () throw () [inline]`

Default Constructor.

6.161.1.2 `decaf::security::cert::CertificateNotYetValidException::CertificateNotYetValidException (const Exception & ex) throw () [inline]`

Conversion Constructor from some other Exception.

Parameters

ex An exception that should become this type of Exception

6.161.1.3 `decaf::security::cert::CertificateNotYetValidException::CertificateNotYetValidException (const CertificateNotYetValidException & ex) throw () [inline]`

Copy Constructor.

Parameters

ex An exception that should become this type of Exception

6.161.1.4 `decaf::security::cert::CertificateNotYetValidException::CertificateNotYetValidException (const char * file, const int lineNumber, const char * msg, ...) throw ()` [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file name where exception occurs

lineNumber line number where the exception occurred.

msg message to report

... list of primitives that are formatted into the message

6.161.1.5 `virtual decaf::security::cert::CertificateNotYetValidException::~~CertificateNotYetValidException () throw ()` [inline, virtual]

6.161.2 Member Function Documentation

6.161.2.1 `virtual CertificateNotYetValidException* decaf::security::cert::CertificateNotYetValidException::clone () const` [inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

A deep copy of this exception.

Reimplemented from `decaf::security::cert::CertificateException` (p.908).

The documentation for this class was generated from the following file:

- `src/main/decaf/security/cert/CertificateNotYetValidException.h`

6.162 decaf::security::cert::CertificateParsingException Class Reference

```
#include <src/main/decaf/security/cert/CertificateParsingException.h>
```

Inheritance diagram for `decaf::security::cert::CertificateParsingException`:

Public Member Functions

- **CertificateParsingException** () throw ()
Default Constructor.
- **CertificateParsingException** (const Exception &ex) throw ()
Conversion Constructor from some other Exception.
- **CertificateParsingException** (const **CertificateParsingException** &ex) throw ()
Copy Constructor.
- **CertificateParsingException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **CertificateParsingException** * clone () const
Clones this exception.
- virtual ~**CertificateParsingException** () throw ()

6.162.1 Constructor & Destructor Documentation

6.162.1.1 decaf::security::cert::CertificateParsingException::CertificateParsingException () throw () [inline]

Default Constructor.

6.162.1.2 decaf::security::cert::CertificateParsingException::CertificateParsingException (const Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

Parameters

ex An exception that should become this type of Exception

6.162.1.3 decaf::security::cert::CertificateParsingException::CertificateParsingException (const CertificateParsingException & ex) throw () [inline]

Copy Constructor.

Parameters

ex An exception that should become this type of Exception

6.162.1.4 `decaf::security::cert::CertificateParsingException::CertificateParsingException`
 (`const char * file`, `const int lineNumber`, `const char * msg`, ...)
 throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file name where exception occurs

lineNumber line number where the exception occurred.

msg message to report

... list of primitives that are formatted into the message

6.162.1.5 `virtual`
`decaf::security::cert::CertificateParsingException::~~CertificateParsingException`
 () throw () [inline, virtual]

6.162.2 Member Function Documentation

6.162.2.1 `virtual CertificateParsingException* de-`
`caf::security::cert::CertificateParsingException::clone (`
`) const` [inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

A deep copy of this exception.

Reimplemented from `decaf::security::cert::CertificateException` (p.908).

The documentation for this class was generated from the following file:

- `src/main/decaf/security/cert/CertificateParsingException.h`

6.163 decaf::lang::Character Class Reference

```
#include <src/main/decaf/lang/Character.h>
```

Inheritance diagram for `decaf::lang::Character`:

Public Member Functions

- `Character` (char value)

- virtual int **compareTo** (const **Character** &c) const
*Compares this **Character** (p. 914) instance with another.*
- virtual bool **operator==** (const **Character** &c) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const **Character** &c) const
Compares this object to another and returns true if this object is considered to be less than the one passed.
- virtual int **compareTo** (const char &c) const
*Compares this **Character** (p. 914) instance with a char type.*
- virtual bool **operator==** (const char &c) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const char &c) const
Compares this object to another and returns true if this object is considered to be less than the one passed.
- bool **equals** (const **Character** &c) const
- bool **equals** (const char &c) const
- std::string **toString** () const
- virtual double **doubleValue** () const
Answers the double value which the receiver represents.
- virtual float **floatValue** () const
Answers the float value which the receiver represents.
- virtual unsigned char **byteValue** () const
Answers the byte value which the receiver represents.
- virtual short **shortValue** () const
Answers the short value which the receiver represents.
- virtual int **intValue** () const
Answers the int value which the receiver represents.
- virtual long long **longValue** () const
Answers the long value which the receiver represents.

Static Public Member Functions

- static **Character** **valueOf** (char value)
*Returns a **Character** (p. 914) instance representing the specified char value.*
- static bool **isWhitespace** (char c)
Indicates whether or not the given character is considered whitespace.

- static bool **isDigit** (char c)
Indicates whether or not the given character is a digit.
- static bool **isLowerCase** (char c)
Indicates whether or not the given character is a lower case character.
- static bool **isUpperCase** (char c)
Indicates whether or not the given character is a upper case character.
- static bool **isLetter** (char c)
Indicates whether or not the given character is a letter.
- static bool **isLetterOrDigit** (char c)
Indicates whether or not the given character is either a letter or a digit.
- static bool **isISOControl** (char c)
Answers whether the character is an ISO control character, which is a char that lays in the range of 0 to 1f and 7f to 9f.
- static int **digit** (char c, int radix)
Returns the numeric value of the character ch in the specified radix.

Static Public Attributes

- static const int **MIN_RADIX** = 2
The minimum radix available for conversion to and from strings.
- static const int **MAX_RADIX** = 36
The maximum radix available for conversion to and from strings.
- static const char **MIN_VALUE** = (char)0x7F
The minimum value that a signed char can take on.
- static const char **MAX_VALUE** = (char)0x80
The maximum value that a signed char can take on.
- static const int **SIZE** = 8
The size of the primitive character in bits.

6.163.1 Constructor & Destructor Documentation

6.163.1.1 decaf::lang::Character::Character (char value)

Parameters

value - char to wrap.

6.163.2 Member Function Documentation

6.163.2.1 `virtual unsigned char decaf::lang::Character::byteValue () const`
`[inline, virtual]`

Answers the byte value which the receiver represents.

Returns

int the value of the receiver.

6.163.2.2 `virtual int decaf::lang::Character::compareTo (const char & c) const`
`[inline, virtual]`

Compares this **Character** (p. 914) instance with a char type.

Parameters

c - the char instance to be compared

Returns

zero if this object represents the same char value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

Implements **decaf::lang::Comparable**< **char** > (p. 1010).

6.163.2.3 `virtual int decaf::lang::Character::compareTo (const Character & c) const`
`[inline, virtual]`

Compares this **Character** (p. 914) instance with another.

Parameters

c - the **Character** (p. 914) instance to be compared

Returns

zero if this object represents the same char value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

6.163.2.4 `static int decaf::lang::Character::digit (char c, int radix) [static]`

Returns the numeric value of the character *ch* in the specified radix.

If the radix is not in the range `MIN_RADIX <= radix <= MAX_RADIX` or if the value of *ch* is not a valid digit in the specified radix, -1 is returned. A character is a valid digit if at least one of the following is true:

* The method `isDigit` is true of the character and the single-character decomposition is less than the specified radix. In this case the decimal digit value is returned. * The character is one of the

uppercase Latin letters 'A' through 'Z' and its code is less than $\text{radix} + 'A' - 10$. In this case, $\text{ch} - 'A' + 10$ is returned. * The character is one of the lowercase Latin letters 'a' through 'z' and its code is less than $\text{radix} + 'a' - 10$. In this case, $\text{ch} - 'a' + 10$ is returned.

Parameters

c - the char to be converted
radix - the radix of the number

Returns

the numeric value of the number represented in the given radix

6.163.2.5 `virtual double decaf::lang::Character::doubleValue () const [inline, virtual]`

Answers the double value which the receiver represents.

Returns

double the value of the receiver.

6.163.2.6 `bool decaf::lang::Character::equals (const char & c) const [inline, virtual]`

Returns

true if the two Characters have the same value.

Implements `decaf::lang::Comparable< char >` (p.1010).

6.163.2.7 `bool decaf::lang::Character::equals (const Character & c) const [inline]`

Returns

true if the two **Character** (p.914) Objects have the same value.

6.163.2.8 `virtual float decaf::lang::Character::floatValue () const [inline, virtual]`

Answers the float value which the receiver represents.

Returns

float the value of the receiver.

6.163.2.9 `virtual int decaf::lang::Character::intValue () const [inline, virtual]`

Answers the int value which the receiver represents.

Returns

int the value of the receiver.

6.163.2.10 `static bool decaf::lang::Character::isDigit (char c) [inline, static]`

Indicates whether or not the given character is a digit.

6.163.2.11 `static bool decaf::lang::Character::isISOControl (char c) [inline, static]`

Answers whether the character is an ISO control character, which is a char that lays in the range of 0 to 1f and 7f to 9f.

Parameters

`c` - the character, including supplementary characters

Returns

true if the char is an ISO control character

6.163.2.12 `static bool decaf::lang::Character::isLetter (char c) [inline, static]`

Indicates whether or not the given character is a letter.

6.163.2.13 `static bool decaf::lang::Character::isLetterOrDigit (char c) [inline, static]`

Indicates whether or not the given character is either a letter or a digit.

6.163.2.14 `static bool decaf::lang::Character::isLowerCase (char c) [inline, static]`

Indicates whether or not the given character is a lower case character.

6.163.2.15 `static bool decaf::lang::Character::isUpperCase (char c) [inline, static]`

Indicates whether or not the given character is a upper case character.

6.163.2.16 `static bool decaf::lang::Character::isWhitespace (char c) [inline, static]`

Indicates whether or not the given character is considered whitespace.

6.163.2.17 `virtual long long decaf::lang::Character::longValue () const [inline, virtual]`

Answers the long value which the receiver represents.

Returns

long the value of the receiver.

6.163.2.18 `virtual bool decaf::lang::Character::operator< (const Character & c) const [inline, virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

Parameters

`c` - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

6.163.2.19 `virtual bool decaf::lang::Character::operator< (const char & c) const [inline, virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

Parameters

`c` - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

Implements `decaf::lang::Comparable< char >` (p. 1011).

6.163.2.20 `virtual bool decaf::lang::Character::operator== (const Character & c) const [inline, virtual]`

Compares equality between this object and the one passed.

Parameters

`c` - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

6.163.2.21 `virtual bool decaf::lang::Character::operator== (const char & c) const [inline, virtual]`

Compares equality between this object and the one passed.

Parameters

`c` - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable< char >** (p. 1011).

6.163.2.22 `virtual short decaf::lang::Character::shortValue () const [inline, virtual]`

Answers the short value which the receiver represents.

Returns

int the value of the receiver.

6.163.2.23 `std::string decaf::lang::Character::toString () const`

Returns

this **Character** (p. 914) Object as a String Representation

6.163.2.24 `static Character decaf::lang::Character::valueOf (char value) [inline, static]`

Returns a **Character** (p. 914) instance representing the specified char value.

Parameters

value - the primitive char to wrap.

Returns

a new Charactor instance that wraps this value.

6.163.3 Field Documentation

6.163.3.1 `const int decaf::lang::Character::MAX_RADIX = 36 [static]`

The maximum radix available for conversion to and from strings.

6.163.3.2 `const char decaf::lang::Character::MAX_VALUE = (char)0x80 [static]`

The maximum value that a signed char can take on.

6.163.3.3 `const int decaf::lang::Character::MIN_RADIX = 2 [static]`

The minimum radix available for conversion to and from strings.

6.163.3.4 `const char decaf::lang::Character::MIN_VALUE = (char)0x7F [static]`

The minimum value that a signed char can take on.

6.163.3.5 const int decaf::lang::Character::SIZE = 8 [static]

The size of the primitive character in bits.

The documentation for this class was generated from the following file:

- src/main/decaf/lang/Character.h

6.164 decaf::internal::nio::CharArrayBuffer Class Reference

```
#include <src/main/decaf/internal/nio/CharArrayBuffer.h>
```

Inheritance diagram for decaf::internal::nio::CharArrayBuffer:

Public Member Functions

- **CharArrayBuffer** (std::size_t capacity, bool **readOnly**=false)
*Creates a **CharArrayBuffer** (p. 922) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*
- **CharArrayBuffer** (char *array, std::size_t **offset**, std::size_t capacity, bool **readOnly**=false) throw (decaf::lang::exceptions::NullPointerException)
*Creates a **CharArrayBuffer** (p. 922) object that wraps the given array.*
- **CharArrayBuffer** (ByteArrayPerspective &array, std::size_t **offset**, std::size_t length, bool **readOnly**=false) throw (decaf::lang::exceptions::IndexOutOfBoundsException)
*Creates a byte buffer that wraps the passed **ByteArrayPerspective** (p. 843) and start at the given offset.*
- **CharArrayBuffer** (const **CharArrayBuffer** &other)
*Create a **CharArrayBuffer** (p. 922) that mirrors this one, meaning it shares a reference to this buffers **ByteArrayPerspective** (p. 843) and when changes are made to that data it is reflected in both.*
- virtual ~**CharArrayBuffer** ()
- virtual char * **array** () throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException)
Returns the character array that backs this buffer (optional operation).
- virtual std::size_t **arrayOffset** () throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException)
Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).
- virtual CharBuffer * **asReadOnlyBuffer** () const
Creates a new, read-only char buffer that shares this buffer's content.
- virtual CharBuffer & **compact** () throw (decaf::nio::ReadOnlyBufferException)
Compacts this buffer.

- virtual CharBuffer * **duplicate** ()
Creates a new char buffer that shares this buffer's content.
- virtual char **get** () throw (decaf::nio::BufferUnderflowException)
Relative get method.
- virtual char **get** (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)
Absolute get method.
- virtual bool **hasArray** () const
Tells whether or not this buffer is backed by an accessible char array.
- virtual bool **isReadOnly** () const
Tells whether or not this buffer is read-only.
- virtual CharBuffer & **put** (char value) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException)
Writes the given char into this buffer at the current position, and then increments the position.
- virtual CharBuffer & **put** (std::size_t index, char value) throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException)
Writes the given char into this buffer at the given index.
- virtual CharBuffer * **slice** () const
Creates a new CharBuffer whose content is a shared subsequence of this buffer's content.
- virtual lang::CharSequence * **subSequence** (std::size_t start, std::size_t end) const throw (decaf::lang::exceptions::IndexOutOfBoundsException)
Creates a new character buffer that represents the specified subsequence of this buffer, relative to the current position.

Protected Member Functions

- virtual void **setReadOnly** (bool value)
*Sets this **ByteBuffer** (p. 806) as Read-Only.*

Protected Attributes

- bool **readOnly**
- internal::nio::ByteBufferPerspective * **__array**
- std::size_t **offset**

6.164.1 Constructor & Destructor Documentation

6.164.1.1 decaf::internal::nio::CharArrayBuffer::CharArrayBuffer (std::size_t capacity, bool readOnly = false)

Creates a **CharArrayBuffer** (p. 922) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.

The array is initially created with all elements initialized to zero.

Parameters

capacity - size of the array, this is the limit we read and write to.

readOnly - should this buffer be read-only, default as false

6.164.1.2 decaf::internal::nio::CharArrayBuffer::CharArrayBuffer (char * array, std::size_t offset, std::size_t capacity, bool readOnly = false) throw (decaf::lang::exceptions::NullPointerException)

Creates a **CharArrayBuffer** (p. 922) object that wraps the given array.

If the own flag is set then it will delete this array when this object is deleted.

Parameters

array - array to wrap

offset - the position that is this buffers start pos.

capacity - size of the array, this is the limit we read and write to.

readOnly - should this buffer be read-only, default as false

Exceptions

NullPointerException if buffer is NULL

6.164.1.3 decaf::internal::nio::CharArrayBuffer::CharArrayBuffer (ByteArrayPerspective & array, std::size_t offset, std::size_t length, bool readOnly = false) throw (decaf::lang::exceptions::IndexOutOfBoundsException)

Creates a byte buffer that wraps the passed **ByteArrayPerspective** (p. 843) and start at the given offset.

The capacity and limit of the new **CharArrayBuffer** (p. 922) will be that of the remaining capacity of the passed buffer.

Parameters

array - the **ByteArrayPerspective** (p. 843) to wrap

offset - the offset into array where the buffer starts

length - the length of the array we are wrapping or limit.

readOnly - is this a readOnly buffer.

Exceptions

IndexOutOfBoundsException if offset is greater than array capacity.

6.164.1.4 `decaf::internal::nio::CharArrayBuffer::CharArrayBuffer (const CharArrayBuffer & other)`

Create a `CharArrayBuffer` (p. 922) that mirrors this one, meaning it shares a reference to this buffers `ByteArrayPerspective` (p. 843) and when changes are made to that data it is reflected in both.

Parameters

other - the `CharArrayBuffer` (p. 922) this one is to mirror.

6.164.1.5 `virtual decaf::internal::nio::CharArrayBuffer::~~CharArrayBuffer ()` [virtual]

6.164.2 Member Function Documentation

6.164.2.1 `virtual char* decaf::internal::nio::CharArrayBuffer::array ()` `throw (decaf::lang::exceptions::UnsupportedOperationException,` `decaf::nio::ReadOnlyBufferException)` [virtual]

Returns the character array that backs this buffer (optional operation).

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

the array that backs this Buffer

Exceptions

ReadOnlyBufferException if this Buffer is read only.

UnsupportedOperationException if the underlying store has no array.

Implements `decaf::nio::CharBuffer` (p. 935).

6.164.2.2 `virtual std::size_t decaf::internal::nio::CharArrayBuffer::arrayOffset ()` `throw (decaf::lang::exceptions::UnsupportedOperationException,` `decaf::nio::ReadOnlyBufferException)` [virtual]

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

The offset into the backing array where index zero starts.

Exceptions

ReadOnlyBufferException if this Buffer is read only.

UnsupportedOperationException if the underlying store has no array.

Implements **decaf::nio::CharBuffer** (p. 936).

6.164.2.3 virtual CharBuffer* decaf::internal::nio::CharArrayBuffer::asReadOnlyBuffer () const [virtual]

Creates a new, read-only char buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns

The new, read-only char buffer which the caller then owns.

Implements **decaf::nio::CharBuffer** (p. 936).

6.164.2.4 virtual CharBuffer& decaf::internal::nio::CharArrayBuffer::compact () throw (decaf::nio::ReadOnlyBufferException) [virtual]

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index $p = \text{position}()$ (p. 746) is copied to index zero, the byte at index $p + 1$ is copied to index one, and so forth until the byte at index $\text{limit}()$ (p. 745) - 1 is copied to index $n = \text{limit}()$ (p. 745) - 1 - p . The buffer's position is then set to $n+1$ and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns

a reference to this CharBuffer

Exceptions

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::CharBuffer** (p. 937).

6.164.2.5 virtual CharBuffer* decaf::internal::nio::CharArrayBuffer::duplicate () [virtual]

Creates a new char buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

a new char Buffer which the caller owns.

Implements **decaf::nio::CharBuffer** (p. 938).

6.164.2.6 `virtual char decaf::internal::nio::CharArrayBuffer::get () throw (decaf::nio::BufferUnderflowException) [virtual]`

Relative get method.

Reads the character at this buffer's current position, and then increments the position.

Returns

the char at the current position

Exceptions

BufferUnderflowException if there no more data to return

Implements **decaf::nio::CharBuffer** (p. 938).

6.164.2.7 `virtual char decaf::internal::nio::CharArrayBuffer::get (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException) [virtual]`

Absolute get method.

Reads the char at the given index.

Parameters

index - the index in the Buffer where the char is to be read

Returns

the char that is located at the given index

Exceptions

IndexOutOfBoundsException - If index is not smaller than the buffer's limit

Implements **decaf::nio::CharBuffer** (p. 938).

6.164.2.8 `virtual bool decaf::internal::nio::CharArrayBuffer::hasArray () const [inline, virtual]`

Tells whether or not this buffer is backed by an accessible char array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Sub-classes should override this method if they do not have a backing array as this class always returns true.

Returns

true if, and only if, this buffer is backed by an array and is not read-only

Implements **decaf::nio::CharBuffer** (p. 940).

6.164.2.9 `virtual bool decaf::internal::nio::CharArrayBuffer::isReadOnly () const`
[inline, virtual]

Tells whether or not this buffer is read-only.

Returns

true if, and only if, this buffer is read-only

6.164.2.10 `virtual CharBuffer& decaf::internal::nio::CharArrayBuffer::put`
`(char value) throw (decaf::nio::BufferOverflowException,`
`decaf::nio::ReadOnlyBufferException) [virtual]`

Writes the given char into this buffer at the current position, and then increments the position.

Parameters

value - the char value to be written

Returns

a reference to this buffer

Exceptions

BufferOverflowException - If this buffer's current position is not smaller than its limit

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::CharBuffer** (p. 942).

6.164.2.11 `virtual CharBuffer& decaf::internal::nio::CharArrayBuffer::put`
`(std::size_t index, char value) throw (`
`decaf::lang::exceptions::IndexOutOfBoundsException,`
`decaf::nio::ReadOnlyBufferException) [virtual]`

Writes the given char into this buffer at the given index.

Parameters

index - position in the Buffer to write the data

value - the char to write.

Returns

a reference to this buffer

Exceptions

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::CharBuffer** (p. 942).

6.164.2.12 **virtual void decaf::internal::nio::CharArrayBuffer::setReadOnly (bool value)** [inline, protected, virtual]

Sets this **ByteBuffer** (p. 806) as Read-Only.

Parameters

value - true if this buffer is to be read-only.

6.164.2.13 **virtual CharBuffer* decaf::internal::nio::CharArrayBuffer::slice ()**
const [virtual]

Creates a new CharBuffer whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the newly create CharBuffer which the caller owns.

Implements **decaf::nio::CharBuffer** (p. 944).

6.164.2.14 **virtual lang::CharSequence* decaf::internal::nio::CharArrayBuffer::subSequence (std::size_t start, std::size_t end) const throw (decaf::lang::exceptions::IndexOutOfBoundsException)** [virtual]

Creates a new character buffer that represents the specified subsequence of this buffer, relative to the current position.

The new buffer will share this buffer's content; that is, if the content of this buffer is mutable then modifications to one buffer will cause the other to be modified. The new buffer's capacity will be that of this buffer, its position will be **position()** (p. 746) + start, and its limit will be **position()** (p. 746) + end. The new Buffer will be read-only if, and only if, this buffer is read-only.

Parameters

start - The index, relative to the current position, of the first character in the subsequence; must be non-negative and no larger than **remaining()** (p. 746)

end - The index, relative to the current position, of the character following the last character in the subsequence; must be no smaller than start and no larger than **remaining()** (p. 746)

Returns

The new character buffer, caller owns

Exceptions

IndexOutOfBoundsException - If the preconditions on start and end fail

Implements **decaf::nio::CharBuffer** (p. 945).

6.164.3 Field Documentation

6.164.3.1 **internal::nio::ByteArrayPerspective***
decaf::internal::nio::CharArrayBuffer::_array [protected]

6.164.3.2 **std::size_t decaf::internal::nio::CharArrayBuffer::offset** [protected]

6.164.3.3 **bool decaf::internal::nio::CharArrayBuffer::readOnly** [protected]

The documentation for this class was generated from the following file:

- src/main/decaf/internal/nio/CharArrayBuffer.h

6.165 decaf::nio::CharBuffer Class Reference

This class defines four categories of operations upon character buffers:

```
#include <src/main/decaf/nio/CharBuffer.h>
```

Inheritance diagram for decaf::nio::CharBuffer:

Public Member Functions

- virtual **~CharBuffer** ()
- virtual std::string **toString** () const
- **CharBuffer & append** (char value) throw (BufferOverflowException, ReadOnlyBufferException)
Appends the specified character to this buffer.
- **CharBuffer & append** (const lang::CharSequence *value) throw (BufferOverflowException, ReadOnlyBufferException)
Appends the specified character sequence to this buffer.
- **CharBuffer & append** (const lang::CharSequence *value, std::size_t start, std::size_t end) throw (decaf::lang::exceptions::IndexOutOfBoundsException, BufferOverflowException, ReadOnlyBufferException)

Appends a subsequence of the specified character sequence to this buffer. If value is Null the the string "null" is appended to the buffer.

- virtual char * **array** ()=0 throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException)

Returns the character array that backs this buffer (optional operation).

- virtual std::size_t **arrayOffset** ()=0 throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException)

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

- virtual **CharBuffer** * **asReadOnlyBuffer** () const =0

Creates a new, read-only char buffer that shares this buffer's content.

- char **charAt** (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException)

Reads the character at the given index relative to the current position.

- virtual **CharBuffer** & **compact** ()=0 throw (ReadOnlyBufferException)

Compacts this buffer.

- virtual **CharBuffer** * **duplicate** ()=0

Creates a new char buffer that shares this buffer's content.

- virtual char **get** ()=0 throw (BufferUnderflowException)

Relative get method.

- virtual char **get** (std::size_t index) const =0 throw (lang::exceptions::IndexOutOfBoundsException)

Absolute get method.

- **CharBuffer** & **get** (std::vector< char > buffer) throw (BufferUnderflowException)

Relative bulk get method.

- **CharBuffer** & **get** (char *buffer, std::size_t offset, std::size_t length) throw (BufferUnderflowException, lang::exceptions::NullPointerException)

Relative bulk get method.

- virtual bool **hasArray** () const =0

Tells whether or not this buffer is backed by an accessible char array.

- std::size_t **length** () const

Returns the length of this character buffer.

- **CharBuffer** & **put** (**CharBuffer** &src) throw (BufferOverflowException, ReadOnlyBufferException, lang::exceptions::IllegalArgumentException)

This method transfers the chars remaining in the given source buffer into this buffer.

- **CharBuffer** & **put** (const char *buffer, std::size_t offset, std::size_t length) throw (BufferOverflowException, ReadOnlyBufferException, lang::exceptions::NullPointerException)

This method transfers chars into this buffer from the given source array.

- **CharBuffer** & **put** (std::vector< char > &buffer) throw (BufferOverflowException, ReadOnlyBufferException)

This method transfers the entire content of the given source char array into this buffer.

- virtual **CharBuffer** & **put** (char value)=0 throw (BufferOverflowException, ReadOnlyBufferException)

Writes the given char into this buffer at the current position, and then increments the position.

- virtual **CharBuffer** & **put** (std::size_t index, char value)=0 throw (lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException)

Writes the given char into this buffer at the given index.

- **CharBuffer** & **put** (const std::string &src, std::size_t start, std::size_t end) throw (BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IndexOutOfBoundsException)

Relative bulk put method (optional operation).

- **CharBuffer** & **put** (const std::string &src) throw (BufferOverflowException, ReadOnlyBufferException)

Relative bulk put method (optional operation).

- virtual std::size_t **read** (**CharBuffer** *target) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, ReadOnlyBufferException)

Attempts to read characters into the specified character buffer.

- virtual lang::CharSequence * **subSequence** (std::size_t start, std::size_t end) const =0 throw (decaf::lang::exceptions::IndexOutOfBoundsException)

Creates a new character buffer that represents the specified subsequence of this buffer, relative to the current position.

- virtual **CharBuffer** * **slice** () const =0

*Creates a new **CharBuffer** (p. 930) whose content is a shared subsequence of this buffer's content.*

- virtual int **compareTo** (const **CharBuffer** &value) const

Compares this object with the specified object for order.

- virtual bool **equals** (const **CharBuffer** &value) const

- virtual bool **operator==** (const **CharBuffer** &value) const

Compares equality between this object and the one passed.

- virtual bool **operator<** (const **CharBuffer** &value) const

Compares this object to another and returns true if this object is considered to be less than the one passed.

Static Public Member Functions

- static **CharBuffer** * **allocate** (std::size_t capacity)
Allocates a new character buffer.
- static **CharBuffer** * **wrap** (char *array, std::size_t offset, std::size_t length) throw (lang::exceptions::NullPointerException)
*Wraps the passed buffer with a new **CharBuffer** (p. 930).*
- static **CharBuffer** * **wrap** (std::vector< char > &buffer)
*Wraps the passed STL char Vector in a **CharBuffer** (p. 930).*

Protected Member Functions

- **CharBuffer** (std::size_t capacity)
*Creates a **CharBuffer** (p. 930) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*

6.165.1 Detailed Description

This class defines four categories of operations upon character buffers: o Absolute and relative get and put methods that read and write single characters; o Relative bulk get methods that transfer contiguous sequences of characters from this buffer into an array; and o Relative bulk put methods that transfer contiguous sequences of characters from a character array, a string, or some other character buffer into this buffer. o Methods for compacting, duplicating, and slicing a character buffer.

Character buffers can be created either by allocation, which allocates space for the buffer's content, by wrapping an existing character array or string into a buffer, or by creating a view of an existing byte buffer

This class implements the CharSequence interface so that character buffers may be used wherever character sequences are accepted, for example in the regular-expression package decaf.util.regex.

Methods in this class that do not otherwise have a value to return are specified to return the buffer upon which they are invoked. This allows method invocations to be chained. The sequence of statements

```
cb.put("text/"); cb.put(subtype); cb.put("; charset="); cb.put(enc);
```

can, for example, be replaced by the single statement

```
cb.put("text/").put(subtype).put("; charset=").put(enc);
```

6.165.2 Constructor & Destructor Documentation

6.165.2.1 decaf::nio::CharBuffer::CharBuffer (std::size_t *capacity*) [protected]

Creates a **CharBuffer** (p. 930) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.

The array is initially created with all elements initialized to zero.

Parameters

capacity - size of the array, this is the limit we read and write to.

6.165.2.2 virtual decaf::nio::CharBuffer::~~CharBuffer () [inline, virtual]

6.165.3 Member Function Documentation

6.165.3.1 static CharBuffer* decaf::nio::CharBuffer::allocate (std::size_t *capacity*) [static]

Allocates a new character buffer.

The new buffer's position will be zero, its limit will be its capacity, and its mark will be undefined. It will have a backing array, and its array offset will be zero.

Parameters

capacity - The size of the Char buffer in chars (1 byte).

Returns

the **CharBuffer** (p. 930) that was allocated, caller owns.

6.165.3.2 CharBuffer& decaf::nio::CharBuffer::append (const lang::CharSequence * *value*, std::size_t *start*, std::size_t *end*) throw (decaf::lang::exceptions::IndexOutOfBoundsException, BufferOverflowException, ReadOnlyBufferException)

Appends a subsequence of the specified character sequence to this buffer If value is Null the the string "null" is appended to the buffer.

Parameters

value - the CharSequence to append.

start - the index to start appending from.

end - the index to append to.

Returns

a reference to this modified **CharBuffer** (p. 930)

Exceptions

BufferOverflowException (p. 771) if there is no more space

ReadOnlyBufferException (p. 2520) if this **Buffer** (p. 741) is read only.

IndexOutOfBoundsException if start > end, or > length of sequence.

6.165.3.3 CharBuffer& decaf::nio::CharBuffer::append (char *value*) throw (BufferOverflowException, ReadOnlyBufferException)

Appends the specified character to this buffer.

Parameters

value - the char to append.

Returns

a reference to this modified **CharBuffer** (p. 930)

Exceptions

BufferOverflowException (p. 771) if there is no more space

ReadOnlyBufferException (p. 2520) if this **Buffer** (p. 741) is read only.

6.165.3.4 CharBuffer& decaf::nio::CharBuffer::append (const lang::CharSequence * *value*) throw (BufferOverflowException, ReadOnlyBufferException)

Appends the specified character sequence to this buffer.

If value is Null the the string "null" is appended to the buffer.

Parameters

value - the CharSequence to append.

Returns

a reference to this modified **CharBuffer** (p. 930)

Exceptions

BufferOverflowException (p. 771) if there is no more space

ReadOnlyBufferException (p. 2520) if this **Buffer** (p. 741) is read only.

6.165.3.5 virtual char* decaf::nio::CharBuffer::array () throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException) [pure virtual]

Returns the character array that backs this buffer (optional operation).

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

the array that backs this **Buffer** (p. 741)

Exceptions

ReadOnlyBufferException (p. 2520) if this **Buffer** (p. 741) is read only.

UnsupportedOperationException if the underlying store has no array.

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 925).

6.165.3.6 `virtual std::size_t decaf::nio::CharBuffer::arrayOffset ()
throw (decaf::lang::exceptions::UnsupportedOperationException,
ReadOnlyBufferException) [pure virtual]`

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

The offset into the backing array where index zero starts.

Exceptions

ReadOnlyBufferException (p. 2520) if this **Buffer** (p. 741) is read only.

UnsupportedOperationException if the underlying store has no array.

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 925).

6.165.3.7 `virtual CharBuffer* decaf::nio::CharBuffer::asReadOnlyBuffer () const
[pure virtual]`

Creates a new, read-only char buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the `duplicate` method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns

The new, read-only char buffer which the caller then owns.

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 926).

6.165.3.8 `char decaf::nio::CharBuffer::charAt (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException)`

Reads the character at the given index relative to the current position.

Parameters

index - The index of the character to be read relative to position

Returns

The character at index **position()** (p. 746) + index

Exceptions

IndexOutOfBoundsException

6.165.3.9 `virtual CharBuffer& decaf::nio::CharBuffer::compact () throw (ReadOnlyBufferException)` [pure virtual]

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index $p = \mathbf{position}()$ (p. 746) is copied to index zero, the byte at index $p + 1$ is copied to index one, and so forth until the byte at index $\mathbf{limit}()$ (p. 745) - 1 is copied to index $n = \mathbf{limit}()$ (p. 745) - 1 - p . The buffer's position is then set to $n+1$ and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns

a reference to this **CharBuffer** (p. 930)

Exceptions

ReadOnlyBufferException (p. 2520) - If this buffer is read-only

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 926).

6.165.3.10 `virtual int decaf::nio::CharBuffer::compareTo (const CharBuffer & value) const` [virtual]

Compares this object with the specified object for order.

Returns a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

Parameters

value - the Object to be compared.

Returns

a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

6.165.3.11 `virtual CharBuffer* decaf::nio::CharBuffer::duplicate () [pure virtual]`

Creates a new char buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

a new char **Buffer** (p. 741) which the caller owns.

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 926).

6.165.3.12 `virtual bool decaf::nio::CharBuffer::equals (const CharBuffer & value) const [virtual]`

Returns

true if this value is considered equal to the passed value.

6.165.3.13 `virtual char decaf::nio::CharBuffer::get () throw (BufferUnderflowException) [pure virtual]`

Relative get method.

Reads the character at this buffer's current position, and then increments the position.

Returns

the char at the current position

Exceptions

BufferUnderflowException (p. 774) if there no more data to return

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 927).

6.165.3.14 `virtual char decaf::nio::CharBuffer::get (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException) [pure virtual]`

Absolute get method.

Reads the char at the given index.

Parameters

index - the index in the **Buffer** (p. 741) where the char is to be read

Returns

the char that is located at the given index

Exceptions

IndexOutOfBoundsException - If index is not smaller than the buffer's limit

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 927).

6.165.3.15 **CharBuffer& decaf::nio::CharBuffer::get (std::vector< char > *buffer*)**
throw (BufferUnderflowException)

Relative bulk get method.

This method transfers chars from this buffer into the given destination vector. An invocation of this method of the form `src.get(a)` behaves in exactly the same way as the invocation. The vector must be sized to the amount of data that is to be read, that is to say, the caller should call `buffer.resize(N)` before calling this get method.

Returns

a reference to this **CharBuffer** (p. 930)

Exceptions

BufferUnderflowException (p. 774) - If there are fewer than length chars remaining in this buffer

6.165.3.16 **CharBuffer& decaf::nio::CharBuffer::get (char * *buffer*, std::size_t**
***offset*, std::size_t *length*) throw (BufferUnderflowException,**
lang::exceptions::NullPointerException)

Relative bulk get method.

This method transfers chars from this buffer into the given destination array. If there are fewer chars remaining in the buffer than are required to satisfy the request, that is, if `length > remaining()` (p. 746), then no bytes are transferred and a **BufferUnderflowException** (p. 774) is thrown.

Otherwise, this method copies length chars from this buffer into the given array, starting at the current position of this buffer and at the given offset in the array. The position of this buffer is then incremented by length.

Parameters

buffer - pointer to an allocated buffer to fill

offset - position in the buffer to start filling

length - amount of data to put in the passed buffer

Returns

a reference to this **Buffer** (p. 741)

Exceptions

BufferUnderflowException (p. 774) - If there are fewer than length chars remaining in this buffer

NullPointerException if the passed buffer is null.

6.165.3.17 `virtual bool decaf::nio::CharBuffer::hasArray () const [pure virtual]`

Tells whether or not this buffer is backed by an accessible char array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Sub-classes should override this method if they do not have a backing array as this class always returns true.

Returns

true if, and only if, this buffer is backed by an array and is not read-only

Implemented in `decaf::internal::nio::CharArrayBuffer` (p. 927).

6.165.3.18 `std::size_t decaf::nio::CharBuffer::length () const [inline]`

Returns the length of this character buffer.

Returns

the length of this buffer from the position to the limit.

6.165.3.19 `virtual bool decaf::nio::CharBuffer::operator< (const CharBuffer & value) const [virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

Parameters

value - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

6.165.3.20 `virtual bool decaf::nio::CharBuffer::operator== (const CharBuffer & value) const [virtual]`

Compares equality between this object and the one passed.

Parameters

value - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

6.165.3.21 `CharBuffer& decaf::nio::CharBuffer::put (CharBuffer & src
) throw (BufferOverflowException, ReadOnlyBufferException,
 lang::exceptions::IllegalArgumentException)`

This method transfers the chars remaining in the given source buffer into this buffer.

If there are more chars remaining in the source buffer than in this buffer, that is, if `src.remaining() > remaining()` (p. 746), then no chars are transferred and a **BufferOverflowException** (p. 771) is thrown.

Otherwise, this method copies `n = src.remaining()` chars from the given buffer into this buffer, starting at each buffer's current position. The positions of both buffers are then incremented by `n`.

Parameters

src - the buffer to take chars from an place in this one.

Returns

a reference to this buffer

Exceptions

BufferOverflowException (p. 771) - If there is insufficient space in this buffer for the remaining chars in the source buffer

IllegalArgumentException - If the source buffer is this buffer

ReadOnlyBufferException (p. 2520) - If this buffer is read-only

6.165.3.22 `CharBuffer& decaf::nio::CharBuffer::put (const char *
 buffer, std::size_t offset, std::size_t length) throw
 (BufferOverflowException, ReadOnlyBufferException,
 lang::exceptions::NullPointerException)`

This method transfers chars into this buffer from the given source array.

If there are more chars to be copied from the array than remain in this buffer, that is, if `length > remaining()` (p. 746), then no chars are transferred and a **BufferOverflowException** (p. 771) is thrown.

Otherwise, this method copies `length` bytes from the given array into this buffer, starting at the given offset in the array and at the current position of this buffer. The position of this buffer is then incremented by `length`.

Parameters

buffer- The array from which chars are to be read

offset- The offset within the array of the first char to be read;

length - The number of chars to be read from the given array

Returns

a reference to this buffer

Exceptions

BufferOverflowException (p. 771) - If there is insufficient space in this buffer

ReadOnlyBufferException (p. 2520) - If this buffer is read-only
NullPointerException if the passed buffer is null.

6.165.3.23 virtual CharBuffer& decaf::nio::CharBuffer::put (char *value*) throw (BufferOverflowException, ReadOnlyBufferException) [pure virtual]

Writes the given char into this buffer at the current position, and then increments the position.

Parameters

value - the char value to be written

Returns

a reference to this buffer

Exceptions

BufferOverflowException (p. 771) - If this buffer's current position is not smaller than its limit

ReadOnlyBufferException (p. 2520) - If this buffer is read-only

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 928).

6.165.3.24 virtual CharBuffer& decaf::nio::CharBuffer::put (std::size_t *index*, char *value*) throw (lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException) [pure virtual]

Writes the given char into this buffer at the given index.

Parameters

index - position in the **Buffer** (p. 741) to write the data

value - the char to write.

Returns

a reference to this buffer

Exceptions

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException (p. 2520) - If this buffer is read-only

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 928).

6.165.3.25 CharBuffer& decaf::nio::CharBuffer::put (const std::string & *src*, std::size_t *start*, std::size_t *end*) throw (BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IndexOutOfBoundsException)

Relative bulk put method (optional operation).

This method transfers characters from the given string into this buffer. If there are more characters to be copied from the string than remain in this buffer, that is, if `end - start > remaining()` (p. 746), then no characters are transferred and a **BufferOverflowException** (p. 771) is thrown.

Returns

a reference to this buffer

Otherwise, this method copies `n = end - start` characters from the given string into this buffer, starting at the given start index and at the current position of this buffer. The position of this buffer is then incremented by `n`.

Parameters

src - the string to copy from
start - position in *src* to start from
end - the position in *src* to stop at

Returns

a reference to this **CharBuffer** (p. 930)

Exceptions

BufferOverflowException (p. 771) - If this buffer's current position is not
IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.
ReadOnlyBufferException (p. 2520) - If this buffer is read-only

6.165.3.26 CharBuffer& decaf::nio::CharBuffer::put (const std::string & src) throw (BufferOverflowException, ReadOnlyBufferException)

Relative bulk put method (optional operation).

This method transfers the entire content of the given source string into this buffer. An invocation of this method of the form `dst.put(s)` behaves in exactly the same way as the invocation

Parameters

src - the string to copy from

Returns

a reference to this **CharBuffer** (p. 930)

Exceptions

BufferOverflowException (p. 771) - If this buffer's current position is not
ReadOnlyBufferException (p. 2520) - If this buffer is read-only

6.165.3.27 CharBuffer& decaf::nio::CharBuffer::put (std::vector< char > & *buffer*) throw (BufferOverflowException, ReadOnlyBufferException)

This method transfers the entire content of the given source char array into this buffer.

This is the same as calling put(&buffer[0], 0, buffer.size()

Parameters

buffer - The buffer whose contents are copied to this **CharBuffer** (p. 930)

Returns

a reference to this buffer

Exceptions

BufferOverflowException (p. 771) - If there is insufficient space in this buffer

ReadOnlyBufferException (p. 2520) - If this buffer is read-only

6.165.3.28 virtual std::size_t decaf::nio::CharBuffer::read (CharBuffer * *target*) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, ReadOnlyBufferException) [virtual]

Attempts to read characters into the specified character buffer.

The buffer is used as a repository of characters as-is: the only changes made are the results of a put operation. No flipping or rewinding of the buffer is performed.

Parameters

target - the buffer to read characters into

Returns

The number of characters added to the buffer, or string::npos if this source of characters is at its end

Exceptions

NullPointerException - If target is Null

IllegalArgumentException - If target is this

ReadOnlyBufferException (p. 2520) - If this buffer is read-only

6.165.3.29 virtual CharBuffer* decaf::nio::CharBuffer::slice () const [pure virtual]

Creates a new **CharBuffer** (p. 930) whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the newly create **CharBuffer** (p. 930) which the caller owns.

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 929).

6.165.3.30 **virtual lang::CharSequence* decaf::nio::CharBuffer::subSequence**
(std::size_t start, std::size_t end) const throw (
decaf::lang::exceptions::IndexOutOfBoundsException) [pure virtual]

Creates a new character buffer that represents the specified subsequence of this buffer, relative to the current position.

The new buffer will share this buffer's content; that is, if the content of this buffer is mutable then modifications to one buffer will cause the other to be modified. The new buffer's capacity will be that of this buffer, its position will be **position()** (p. 746) + start, and its limit will be **position()** (p. 746) + end. The new **Buffer** (p. 741) will be read-only if, and only if, this buffer is read-only.

Parameters

start - The index, relative to the current position, of the first character in the subsequence; must be non-negative and no larger than **remaining()** (p. 746)

end - The index, relative to the current position, of the character following the last character in the subsequence; must be no smaller than start and no larger than **remaining()** (p. 746)

Returns

The new character buffer, caller owns

Exceptions

IndexOutOfBoundsException - If the preconditions on start and end fail

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 929).

6.165.3.31 **virtual std::string decaf::nio::CharBuffer::toString () const** [virtual]

Returns

a std::string describing this object

6.165.3.32 **static CharBuffer* decaf::nio::CharBuffer::wrap (char ***
array, std::size_t offset, std::size_t length) throw (
lang::exceptions::NullPointerException) [static]

Wraps the passed buffer with a new **CharBuffer** (p. 930).

The new buffer will be backed by the given char array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be array.length, its position will be offset, its limit will be offset + length, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

array - The array that will back the new buffer

offset - The offset of the subarray to be used

length - The length of the subarray to be used

Returns

a new **CharBuffer** (p. 930) that is backed by buffer, caller owns.

6.165.3.33 static CharBuffer* decaf::nio::CharBuffer::wrap (std::vector< char > & buffer) [static]

Wraps the passed STL char Vector in a **CharBuffer** (p. 930).

The new buffer will be backed by the given char array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be buffer.size(), its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

buffer - The vector that will back the new buffer, the vector must have been sized to the desired size already by calling vector.resize(N).

Returns

a new **CharBuffer** (p. 930) that is backed by buffer, caller owns.

The documentation for this class was generated from the following file:

- src/main/decaf/nio/CharBuffer.h

6.166 decaf::lang::CharSequence Class Reference

A **CharSequence** (p. 946) is a readable sequence of char values.

```
#include <src/main/decaf/lang/CharSequence.h>
```

Inheritance diagram for decaf::lang::CharSequence:

Public Member Functions

- virtual **~CharSequence** ()
- virtual std::size_t **length** () const =0
- virtual char **charAt** (std::size_t index) const =0 throw (lang::exceptions::IndexOutOfBoundsException)
Returns the Char at the specified index so long as the index is not greater than the length of the sequence.
- virtual **CharSequence** * **subSequence** (std::size_t start, std::size_t end) const =0 throw (lang::exceptions::IndexOutOfBoundsException)
*Returns a new **CharSequence** (p. 946) that is a subsequence of this sequence.*
- virtual std::string **toString** () const =0

6.166.1 Detailed Description

A **CharSequence** (p. 946) is a readable sequence of char values. This interface provides uniform, read-only access to many different kinds of char sequences.

This interface does not define that a **CharSequence** (p. 946) should implement comparable, it is therefore up to the dervied classes that implement this interface to define equality, which implies that comparison of two CharSequences does not have a contract on equality.

6.166.2 Constructor & Destructor Documentation

6.166.2.1 `virtual decaf::lang::CharSequence::~~CharSequence () [inline, virtual]`

6.166.3 Member Function Documentation

6.166.3.1 `virtual char decaf::lang::CharSequence::charAt (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException) [pure virtual]`

Returns the Char at the specified index so long as the index is not greater than the length of the sequence.

Parameters

index - position to return the char at.

Returns

the char at the given position

Exceptions

IndexOutOfBoundsException if index is > than `length()` (p. 947)

6.166.3.2 `virtual std::size_t decaf::lang::CharSequence::length () const [pure virtual]`

Returns

the length of the underlying character sequence.

6.166.3.3 `virtual CharSequence* decaf::lang::CharSequence::subSequence (std::size_t start, std::size_t end) const throw (lang::exceptions::IndexOutOfBoundsException) [pure virtual]`

Returns a new **CharSequence** (p. 946) that is a subsequence of this sequence.

The subsequence starts with the char value at the specified index and ends with the char value at index `end - 1`. The length (in chars) of the returned sequence is `end - start`, so if `start == end` then an empty sequence is returned.

Parameters

- start* - the start index, inclusive
- end* - the end index, exclusive

Returns

a new **CharSequence** (p. 946)

Exceptions

IndexOutOfBoundsException if start or end > **length()** (p. 947)

6.166.3.4 `virtual std::string decaf::lang::CharSequence::toString () const [pure virtual]`

Returns

the string representation of this **CharSequence** (p. 946)

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**CharSequence.h**

6.167 decaf::lang::exceptions::ClassCastException Class Reference

```
#include <src/main/decaf/lang/exceptions/ClassCastException.h>
```

Inheritance diagram for decaf::lang::exceptions::ClassCastException:

Public Member Functions

- **ClassCastException** () throw ()
Default Constructor.
- **ClassCastException** (const **Exception** &ex) throw ()
*Conversion Constructor from some other **Exception** (p. 1476).*
- **ClassCastException** (const **ClassCastException** &ex) throw ()
Copy Constructor.
- **ClassCastException** (const std::exception *cause) throw ()
Constructor.
- **ClassCastException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.

- **ClassCastException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **ClassCastException** * clone () const
Clones this exception.
- virtual ~**ClassCastException** () throw ()

6.167.1 Constructor & Destructor Documentation

6.167.1.1 decaf::lang::exceptions::ClassCastException::ClassCastException () throw () [inline]

Default Constructor.

6.167.1.2 decaf::lang::exceptions::ClassCastException::ClassCastException (const Exception & ex) throw () [inline]

Conversion Constructor from some other **Exception** (p. 1476).

Parameters

ex The **Exception** (p. 1476) whose data is to be copied into this one.

6.167.1.3 decaf::lang::exceptions::ClassCastException::ClassCastException (const ClassCastException & ex) throw () [inline]

Copy Constructor.

Parameters

ex The **Exception** (p. 1476) whose data is to be copied into this one.

6.167.1.4 decaf::lang::exceptions::ClassCastException::ClassCastException (const std::exception * cause) throw () [inline]

Constructor.

Parameters

cause **Pointer** (p. 2343) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.167.1.5 decaf::lang::exceptions::ClassCastException::ClassCastException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs
lineNumber The line number where the exception occurred.
msg The message to report
... list of primitives that are formatted into the message

6.167.1.6 `decaf::lang::exceptions::ClassCastException::ClassCastException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs
lineNumber The line number where the exception occurred.
cause The exception that was the cause for this one to be thrown.
msg The message to report
... list of primitives that are formatted into the message

6.167.1.7 `virtual decaf::lang::exceptions::ClassCastException::~~ClassCastException () throw () [inline, virtual]`

6.167.2 Member Function Documentation

6.167.2.1 `virtual ClassCastException* decaf::lang::exceptions::ClassCastException::clone () const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

an new **Exception** (p. 1476) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1479).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/exceptions/ClassCastException.h`

6.168 decaf::io::Closeable Class Reference

Interface for a class that implements the close method.

```
#include <src/main/decaf/io/Closeable.h>
```

Inheritance diagram for decaf::io::Closeable:

Public Member Functions

- virtual `~Closeable()`
- virtual void `close()` throw (io::IOException)
Closes this object and deallocates the appropriate resources.

6.168.1 Detailed Description

Interface for a class that implements the close method.

6.168.2 Constructor & Destructor Documentation

6.168.2.1 virtual decaf::io::Closeable::~~Closeable () [inline, virtual]

6.168.3 Member Function Documentation

6.168.3.1 virtual void decaf::io::Closeable::close () throw (io::IOException)
 [pure virtual]

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling close.

Exceptions

IOException (p. 1707) if an error occurs while closing.

Implemented in `activemq::transport::mock::MockTransport` (p. 2229), `decaf::net::BufferedSocket` (p. 757), `decaf::net::TcpSocket` (p. 2961), and `decaf::util::logging::StreamHandler` (p. 2889).

The documentation for this class was generated from the following file:

- `src/main/decaf/io/Closeable.h`

6.169 cms::Closeable Class Reference

Interface for a class that implements the close method.

```
#include <src/main/cms/Closeable.h>
```

Inheritance diagram for cms::Closeable:

Public Member Functions

- virtual `~Closeable ()`
- virtual void `close ()=0 throw (CMSEException)`

Closes this object and deallocates the appropriate resources.

6.169.1 Detailed Description

Interface for a class that implements the close method. A class that implements this interface should release all resources upon the close call and should throw an exception from any methods that require those resources after they have been closed.

Since

1.0

6.169.2 Constructor & Destructor Documentation

6.169.2.1 virtual `cms::Closeable::~~Closeable ()` [inline, virtual]

6.169.3 Member Function Documentation

6.169.3.1 virtual void `cms::Closeable::close () throw (CMSEException)` [pure virtual]

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling close.

Exceptions

CMSEException (p. 960) - If an error occurs while the resource is being closed.

Implemented in `activemq::cmsutil::CachedConsumer` (p. 889), `activemq::cmsutil::CachedProducer` (p. 892), `activemq::cmsutil::PooledSession` (p. 2354), `activemq::commands::ActiveMQTempDestination` (p. 457), `activemq::core::ActiveMQProducer` (p. 369), `cms::Connection` (p. 1053), and `cms::Session` (p. 2667).

The documentation for this class was generated from the following file:

- `src/main/cms/Closeable.h`

6.170 activemq::transport::failover::CloseTransportsTask Class Reference

```
#include <src/main/activemq/transport/failover/CloseTransportsTask.h>
```

Inheritance diagram for `activemq::transport::failover::CloseTransportsTask`:

Public Member Functions

- **CloseTransportsTask** ()
- virtual **~CloseTransportsTask** ()
- void **add** (const **Pointer**< **Transport** > &transport)
*Add a new **Transport** (p. 3066) to close.*
- virtual bool **isPending** () const
This Task is pending if there are transports in the Queue that need to be closed.
- virtual bool **iterate** ()
Return true until all transports have been closed and removed from the queue.

6.170.1 Constructor & Destructor Documentation

- 6.170.1.1** **activemq::transport::failover::CloseTransportsTask::CloseTransportsTask**
 () [inline]
- 6.170.1.2** **virtual**
activemq::transport::failover::CloseTransportsTask::~~CloseTransportsTask
 () [inline, virtual]

6.170.2 Member Function Documentation

- 6.170.2.1** **void activemq::transport::failover::CloseTransportsTask::add** (const
Pointer< **Transport** > & *transport*)

Add a new **Transport** (p. 3066) to close.

- 6.170.2.2** **virtual bool activemq::transport::failover::CloseTransportsTask::isPending**
 () const [virtual]

This Task is pending if there are transports in the Queue that need to be closed.

Returns

true if there is a transport in the queue that needs closed.

Implements **activemq::threads::CompositeTask** (p. 1016).

- 6.170.2.3** **virtual bool activemq::transport::failover::CloseTransportsTask::iterate** (
) [virtual]

Return true until all transports have been closed and removed from the queue.

Implements **activemq::threads::Task** (p. 2956).

The documentation for this class was generated from the following file:

- src/main/activemq/transport/failover/**CloseTransportsTask.h**

6.171 activemq::cmsutil::CmsAccessor Class Reference

Base class for **activemq.cmsutil.CmsTemplate** (p.969) and other CMS-accessing gateway helpers, defining common properties such as the CMS **cms.ConnectionFactory** (p.1103) to operate on.

```
#include <src/main/activemq/cmsutil/CmsAccessor.h>
```

Inheritance diagram for activemq::cmsutil::CmsAccessor:

Public Member Functions

- **CmsAccessor** ()
- virtual **~CmsAccessor** ()
- virtual **ResourceLifecycleManager** * **getResourceLifecycleManager** ()
- virtual const **ResourceLifecycleManager** * **getResourceLifecycleManager** () const
- virtual void **setConnectionFactory** (**cms::ConnectionFactory** *connectionFactory)
Set the ConnectionFactory to use for obtaining CMS Connections.
- virtual const **cms::ConnectionFactory** * **getConnectionFactory** () const
Return the ConnectionFactory that this accessor uses for obtaining CMS Connections.
- virtual **cms::ConnectionFactory** * **getConnectionFactory** ()
Return the ConnectionFactory that this accessor uses for obtaining CMS Connections.
- virtual void **setSessionAcknowledgeMode** (**cms::Session::AcknowledgeMode** sessionAcknowledgeMode)
Set the CMS acknowledgment mode that is used when creating a CMS Session to send a message.
- virtual **cms::Session::AcknowledgeMode** **getSessionAcknowledgeMode** () const
Return the acknowledgment mode for CMS sessions.

Protected Member Functions

- virtual void **init** () throw (cms::CMSException, decaf::lang::exceptions::IllegalStateException)
Initializes this object and prepares it for use.
- virtual void **destroy** () throw (cms::CMSException, decaf::lang::exceptions::IllegalStateException)
Shuts down this object and destroys any allocated resources.
- virtual **cms::Connection** * **createConnection** () throw (cms::CMSException, decaf::lang::exceptions::IllegalStateException)
Create a CMS Connection via this template's ConnectionFactory.
- virtual **cms::Session** * **createSession** (**cms::Connection** *con) throw (cms::CMSException, decaf::lang::exceptions::IllegalStateException)

Create a CMS Session for the given Connection.

- virtual void **checkConnectionFactory** () throw (decaf::lang::exceptions::IllegalStateException)

Verifies that the connection factory is valid.

6.171.1 Detailed Description

Base class for **activemq.cmsutil.CmsTemplate** (p. 969) and other CMS-accessing gateway helpers, defining common properties such as the CMS **cms.ConnectionFactory** (p. 1103) to operate on. The subclass **activemq.cmsutil.CmsDestinationAccessor** (p. 957) adds further, destination-related properties.

Not intended to be used directly.

See also

activemq.cmsutil.CmsDestinationAccessor (p. 957)
activemq.cmsutil.CmsTemplate (p. 969)

6.171.2 Constructor & Destructor Documentation

6.171.2.1 **activemq::cmsutil::CmsAccessor::CmsAccessor** ()

6.171.2.2 **virtual activemq::cmsutil::CmsAccessor::~~CmsAccessor** () [virtual]

6.171.3 Member Function Documentation

6.171.3.1 **virtual void activemq::cmsutil::CmsAccessor::checkConnectionFactory** () throw (decaf::lang::exceptions::IllegalStateException) [protected, virtual]

Verifies that the connection factory is valid.

6.171.3.2 **virtual cms::Connection* activemq::cmsutil::CmsAccessor::createConnection** () throw (cms::CMSException, decaf::lang::exceptions::IllegalStateException) [protected, virtual]

Create a CMS Connection via this template's ConnectionFactory.

Returns

the new CMS Connection

Exceptions

cms::CMSException (p. 960) if thrown by CMS API methods

6.171.3.3 `virtual cms::Session* activemq::cmsutil::CmsAccessor::createSession (cms::Connection * con) throw (cms::CMSException, decaf::lang::exceptions::IllegalStateException) [protected, virtual]`

Create a CMS Session for the given Connection.

Parameters

con the CMS Connection to create a Session for

Returns

the new CMS Session

Exceptions

cms::CMSException (p. 960) if thrown by CMS API methods

6.171.3.4 `virtual void activemq::cmsutil::CmsAccessor::destroy () throw (cms::CMSException, decaf::lang::exceptions::IllegalStateException) [inline, protected, virtual]`

Shuts down this object and destroys any allocated resources.

Reimplemented in `activemq::cmsutil::CmsDestinationAccessor` (p. 959), and `activemq::cmsutil::CmsTemplate` (p. 973).

6.171.3.5 `virtual cms::ConnectionFactory* activemq::cmsutil::CmsAccessor::getConnectionFactory () [inline, virtual]`

Return the ConnectionFactory that this accessor uses for obtaining CMS Connections.

6.171.3.6 `virtual const cms::ConnectionFactory* activemq::cmsutil::CmsAccessor::getConnectionFactory () const [inline, virtual]`

Return the ConnectionFactory that this accessor uses for obtaining CMS Connections.

6.171.3.7 `virtual ResourceLifecycleManager* activemq::cmsutil::CmsAccessor::getResourceLifecycleManager () [inline, virtual]`

6.171.3.8 `virtual const ResourceLifecycleManager* activemq::cmsutil::CmsAccessor::getResourceLifecycleManager () const [inline, virtual]`

6.171.3.9 `virtual cms::Session::AcknowledgeMode activemq::cmsutil::CmsAccessor::getSessionAcknowledgeMode () const [inline, virtual]`

Return the acknowledgment mode for CMS sessions.

Returns

the acknowledgment mode applied by this accessor

6.171.3.10 `virtual void activemq::cmsutil::CmsAccessor::init () throw (cms::CMSException, decaf::lang::exceptions::IllegalStateException) [inline, protected, virtual]`

Initializes this object and prepares it for use.

This should be called before any other methods are called. This version does nothing.

Reimplemented in `activemq::cmsutil::CmsDestinationAccessor` (p.959), and `activemq::cmsutil::CmsTemplate` (p.975).

6.171.3.11 `virtual void activemq::cmsutil::CmsAccessor::setConnectionFactory (cms::ConnectionFactory * connectionFactory) [inline, virtual]`

Set the ConnectionFactory to use for obtaining CMS Connections.

6.171.3.12 `virtual void activemq::cmsutil::CmsAccessor::setSessionAcknowledgeMode (cms::Session::AcknowledgeMode sessionAcknowledgeMode) [inline, virtual]`

Set the CMS acknowledgment mode that is used when creating a CMS Session to send a message.

Default is AUTO_ACKNOWLEDGE.

Parameters

sessionAcknowledgeMode the acknowledgment mode

The documentation for this class was generated from the following file:

- `src/main/activemq/cmsutil/CmsAccessor.h`

6.172 `activemq::cmsutil::CmsDestinationAccessor` Class Reference

Extends the `CmsAccessor` (p.954) to add support for resolving destination names.

```
#include <src/main/activemq/cmsutil/CmsDestinationAccessor.h>
```

Inheritance diagram for `activemq::cmsutil::CmsDestinationAccessor`:

Public Member Functions

- `CmsDestinationAccessor ()`
- `virtual ~CmsDestinationAccessor ()`

- virtual bool **isPubSubDomain** () const
- virtual void **setPubSubDomain** (bool pubSubDomain)
- virtual **DestinationResolver** * **getDestinationResolver** ()
- virtual const **DestinationResolver** * **getDestinationResolver** () const
- virtual void **setDestinationResolver** (**DestinationResolver** *destRes)

Protected Member Functions

- virtual void **init** () throw (cms::CMSEException, decaf::lang::exceptions::IllegalStateException)

Initializes the destination resolver.

- virtual void **destroy** () throw (cms::CMSEException, decaf::lang::exceptions::IllegalStateException)

*Calls **destroy()** (p. 959) on the destination resolver.*

- virtual **cms::Destination** * **resolveDestinationName** (**cms::Session** *session, const std::string &destName) throw (cms::CMSEException, decaf::lang::exceptions::IllegalStateException)

*Resolves the destination via the **DestinationResolver** (p. 1415).*

- virtual void **checkDestinationResolver** () throw (decaf::lang::exceptions::IllegalStateException)

Verifies that the destination resolver is valid.

6.172.1 Detailed Description

Extends the **CmsAccessor** (p. 954) to add support for resolving destination names. Not intended to be used directly.

See also

CmsTemplate (p. 969)
CmsAccessor (p. 954)

6.172.2 Constructor & Destructor Documentation

6.172.2.1 `activemq::cmsutil::CmsDestinationAccessor::CmsDestinationAccessor ()`

6.172.2.2 `virtual
activemq::cmsutil::CmsDestinationAccessor::~~CmsDestinationAccessor ()` [virtual]

6.172.3 Member Function Documentation

6.172.3.1 `virtual void activemq::cmsutil::CmsDestinationAccessor::checkDestinationResolver ()` throw (`decaf::lang::exceptions::IllegalStateException`) [protected, virtual]

Verifies that the destination resolver is valid.

6.172.3.2 `virtual void activemq::cmsutil::CmsDestinationAccessor::destroy ()` throw (`cms::CMSException`, `decaf::lang::exceptions::IllegalStateException`) [protected, virtual]

Calls `destroy()` (p. 959) on the destination resolver.

Reimplemented from `activemq::cmsutil::CmsAccessor` (p. 956).

Reimplemented in `activemq::cmsutil::CmsTemplate` (p. 973).

6.172.3.3 `virtual const DestinationResolver* activemq::cmsutil::CmsDestinationAccessor::getDestinationResolver ()` const [inline, virtual]

6.172.3.4 `virtual DestinationResolver* activemq::cmsutil::CmsDestinationAccessor::getDestinationResolver ()` [inline, virtual]

6.172.3.5 `virtual void activemq::cmsutil::CmsDestinationAccessor::init ()` throw (`cms::CMSException`, `decaf::lang::exceptions::IllegalStateException`) [protected, virtual]

Initializes the destination resolver.

Reimplemented from `activemq::cmsutil::CmsAccessor` (p. 957).

Reimplemented in `activemq::cmsutil::CmsTemplate` (p. 975).

6.172.3.6 virtual bool activemq::cmsutil::CmsDestinationAccessor::isPubSubDomain () const
[inline, virtual]

6.172.3.7 virtual cms::Destination* activemq::cmsutil::CmsDestinationAccessor::resolveDestinationName (cms::Session * *session*, const std::string & *destName*) throw (cms::CMSException, decaf::lang::exceptions::IllegalStateException)
[protected, virtual]

Resolves the destination via the DestinationResolver (p. 1415).

Parameters

session the session

destName the name of the destination.

Returns

the destination

Exceptions

cms::CMSException (p. 960) if resolution failed.

decaf::lang::exceptions::IllegalStateException (p. 1618) if the destination resolver property is NULL.

6.172.3.8 virtual void activemq::cmsutil::CmsDestinationAccessor::setDestinationResolver (DestinationResolver * *destRes*) [inline, virtual]

6.172.3.9 virtual void activemq::cmsutil::CmsDestinationAccessor::setPubSubDomain (bool *pubSubDomain*) [inline, virtual]

Reimplemented in *activemq::cmsutil::CmsTemplate* (p. 980).

Referenced by *activemq::cmsutil::CmsTemplate::setPubSubDomain()*.

The documentation for this class was generated from the following file:

- *src/main/activemq/cmsutil/CmsDestinationAccessor.h*

6.173 cms::CMSException Class Reference

CMS API Exception that is the base for all exceptions thrown from CMS classes.

```
#include <src/main/cms/CMSException.h>
```

Inheritance diagram for cms::CMSException:

Public Member Functions

- **CMSEException** () throw ()
- **CMSEException** (const **CMSEException** &ex) throw ()
- **CMSEException** (const std::string &message, const std::exception *cause) throw ()
- **CMSEException** (const std::string &message, const std::exception *cause, const std::vector< std::pair< std::string, int > > &stackTrace) throw ()
- virtual ~**CMSEException** () throw ()
- virtual std::string **getMessage** () const
Gets the cause of the error.
- virtual const std::exception * **getCause** () const
Gets the exception that caused this one to be thrown, this allows for chaining of exceptions in the case of a method that throws only a particular exception but wishes to allow for the real causal exception to be passed only in case the caller knows about that type of exception and wishes to respond to it.
- virtual std::vector< std::pair< std::string, int > > **getStackTrace** () const
Provides the stack trace for every point where this exception was caught, marked, and rethrown.
- virtual void **setMark** (const char *file, const int lineNumber)
Adds a file/line number to the stack trace.
- virtual void **printStackTrace** () const
Prints the stack trace to std::err.
- virtual void **printStackTrace** (std::ostream &stream) const
Prints the stack trace to the given output stream.
- virtual std::string **getStackTraceString** () const
Gets the stack trace as one contiguous string.
- virtual const char * **what** () const throw ()
*Overloads the std::exception **what**() (p. 963) function to return the cause of the exception.*

6.173.1 Detailed Description

CMS API Exception that is the base for all exceptions thrown from CMS classes. This class represents an error that has occurred in CMS, providers can wrap provider specific exceptions in this class by setting the cause to an instance of a provider specific exception provided it can be cast to an std::exception.

Since the contained cause exception is of type std::exception and the C++ exception class has no clone or copy method defined the contained exception can only be owned by one instance of an **CMSEException** (p. 960). To that end the class hands off the exception to each successive copy so care must be taken when handling **CMSEException** (p. 960) instances.

Since

1.0

6.173.2 Constructor & Destructor Documentation

- 6.173.2.1** cms::CMSException::CMSException () throw ()
- 6.173.2.2** cms::CMSException::CMSException (const CMSException & *ex*) throw ()
- 6.173.2.3** cms::CMSException::CMSException (const std::string & *message*, const std::exception * *cause*) throw ()
- 6.173.2.4** cms::CMSException::CMSException (const std::string & *message*, const std::exception * *cause*, const std::vector< std::pair< std::string, int > > & *stackTrace*) throw ()
- 6.173.2.5** virtual cms::CMSException::~~CMSException () throw () [virtual]

6.173.3 Member Function Documentation

- 6.173.3.1** virtual const std::exception* cms::CMSException::getCause () const [virtual]

Gets the exception that caused this one to be thrown, this allows for chaining of exceptions in the case of a method that throws only a particular exception but wishes to allow for the real causal exception to be passed only in case the caller knows about that type of exception and wishes to respond to it.

Returns

a const pointer reference to the causal exception, if there was no cause associated with this exception then NULL is returned.

- 6.173.3.2** virtual std::string cms::CMSException::getMessage () const [virtual]

Gets the cause of the error.

Returns

string errors message

- 6.173.3.3** virtual std::vector< std::pair< std::string, int> > cms::CMSException::getStackTrace () const [virtual]

Provides the stack trace for every point where this exception was caught, marked, and rethrown.

Returns

vector containing stack trace strings

6.173.3.4 `virtual std::string cms::CMSException::getStackTraceString () const [virtual]`

Gets the stack trace as one contiguous string.

Returns

string with formatted stack trace data

6.173.3.5 `virtual void cms::CMSException::printStackTrace () const [virtual]`

Prints the stack trace to std::err.

6.173.3.6 `virtual void cms::CMSException::printStackTrace (std::ostream & stream) const [virtual]`

Prints the stack trace to the given output stream.

Parameters

stream the target output stream.

6.173.3.7 `virtual void cms::CMSException::setMark (const char * file, const int lineNumber) [virtual]`

Adds a file/line number to the stack trace.

Parameters

file The name of the file calling this method (use `__FILE__`).

lineNumber The line number in the calling file (use `__LINE__`).

6.173.3.8 `virtual const char* cms::CMSException::what () const throw () [virtual]`

Overloads the std::exception `what()` (p. 963) function to return the cause of the exception.

Returns

const char pointer to error message

The documentation for this class was generated from the following file:

- `src/main/cms/CMSException.h`

6.174 activemq::util::CMSExceptionSupport Class Reference

```
#include <src/main/activemq/util/CMSExceptionSupport.h>
```

Public Member Functions

- virtual `~CMSExceptionSupport ()`

Static Public Member Functions

- static `cms::CMSException create (const std::string &msg, const decaf::lang::Exception &cause)`
- static `cms::CMSException create (const decaf::lang::Exception &cause)`
- static `cms::MessageEOFException createMessageEOFException (const decaf::lang::Exception &cause)`
- static `cms::MessageFormatException createMessageFormatException (const decaf::lang::Exception &cause)`

6.174.1 Constructor & Destructor Documentation

- 6.174.1.1** virtual `activemq::util::CMSExceptionSupport::~~CMSExceptionSupport ()` [virtual]

6.174.2 Member Function Documentation

- 6.174.2.1** static `cms::CMSException activemq::util::CMSExceptionSupport::create (const std::string & msg, const decaf::lang::Exception & cause)` [static]
- 6.174.2.2** static `cms::CMSException activemq::util::CMSExceptionSupport::create (const decaf::lang::Exception & cause)` [static]
- 6.174.2.3** static `cms::MessageEOFException activemq::util::CMSExceptionSupport::createMessageEOFException (const decaf::lang::Exception & cause)` [static]
- 6.174.2.4** static `cms::MessageFormatException activemq::util::CMSExceptionSupport::createMessageFormatException (const decaf::lang::Exception & cause)` [static]

Referenced by `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getBooleanProperty()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getByteProperty()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getDoubleProperty()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getFloatProperty()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getIntProperty()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getLongProperty()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getShortProperty()`, and `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getStringProperty()`.

The documentation for this class was generated from the following file:

- `src/main/activemq/util/CMSExceptionSupport.h`

6.175 cms::CMSProperties Class Reference

Interface for a Java-like properties object.

```
#include <src/main/cms/CMSProperties.h>
```

Inheritance diagram for cms::CMSProperties:

Public Member Functions

- virtual `~CMSProperties ()`
- virtual bool `isEmpty ()` const =0
Returns true if the properties object is empty.
- virtual const char * `getProperty (const std::string &name)` const =0
Looks up the value for the given property.
- virtual std::string `getProperty (const std::string &name, const std::string &defaultValue)` const =0
Looks up the value for the given property.
- virtual void `setProperty (const std::string &name, const std::string &value)`=0
Sets the value for a given property.
- virtual bool `hasProperty (const std::string &name)` const =0
Check to see if the Property exists in the set.
- virtual void `remove (const std::string &name)`=0
Removes the property with the given name.
- virtual std::vector< std::pair< std::string, std::string > > `toArray ()` const =0
Method that serializes the contents of the property map to an array.
- virtual void `copy (const CMSProperties *source)`=0
Copies the contents of the given properties object to this one.
- virtual `CMSProperties * clone ()` const =0
Clones this object.
- virtual void `clear ()`=0
Clears all properties from the map.
- virtual std::string `toString ()` const =0
Formats the contents of the Properties Object into a string that can be logged, etc.

6.175.1 Detailed Description

Interface for a Java-like properties object. This is essentially a map of key-value string pairs.

Since

1.1

6.175.2 Constructor & Destructor Documentation

6.175.2.1 `virtual cms::CMSProperties::~~CMSProperties () [inline, virtual]`

6.175.3 Member Function Documentation

6.175.3.1 `virtual void cms::CMSProperties::clear () [pure virtual]`

Clears all properties from the map.

Implemented in `activemq::util::ActiveMQProperties` (p. 376).

6.175.3.2 `virtual CMSProperties* cms::CMSProperties::clone () const [pure virtual]`

Clones this object.

Returns

a replica of this object.

Implemented in `activemq::util::ActiveMQProperties` (p. 376).

6.175.3.3 `virtual void cms::CMSProperties::copy (const CMSProperties * source) [pure virtual]`

Copies the contents of the given properties object to this one.

Parameters

source The source properties object.

6.175.3.4 `virtual const char* cms::CMSProperties::getProperty (const std::string & name) const [pure virtual]`

Looks up the value for the given property.

Parameters

name The name of the property to be looked up.

Returns

the value of the property with the given name, if it exists. If it does not exist, returns NULL.

Implemented in `activemq::util::ActiveMQProperties` (p. 377).

6.175.3.5 `virtual std::string cms::CMSProperties::getProperty (const std::string & name, const std::string & defaultValue) const` [pure virtual]

Looks up the value for the given property.

Parameters

name the name of the property to be looked up.

defaultValue The value to be returned if the given property does not exist.

Returns

The value of the property specified by *name*, if it exists, otherwise the *defaultValue*.

Implemented in `activemq::util::ActiveMQProperties` (p. 377).

6.175.3.6 `virtual bool cms::CMSProperties::hasProperty (const std::string & name) const` [pure virtual]

Check to see if the Property exists in the set.

Parameters

name the name of the property to check

Returns

true if property exists, false otherwise.

Implemented in `activemq::util::ActiveMQProperties` (p. 377).

6.175.3.7 `virtual bool cms::CMSProperties::isEmpty () const` [pure virtual]

Returns true if the properties object is empty.

Returns

true if empty

Implemented in `activemq::util::ActiveMQProperties` (p. 378).

6.175.3.8 `virtual void cms::CMSProperties::remove (const std::string & name)` [pure virtual]

Removes the property with the given name.

Parameters

name the name of the property to be removed.s

Implemented in `activemq::util::ActiveMQProperties` (p. 378).

6.175.3.9 `virtual void cms::CMSProperties::setProperty (const std::string & name, const std::string & value) [pure virtual]`

Sets the value for a given property.

If the property already exists, overwrites the value.

Parameters

name The name of the value to be written.

value The value to be written.

Implemented in `activemq::util::ActiveMQProperties` (p. 378).

6.175.3.10 `virtual std::vector< std::pair< std::string, std::string > > cms::CMSProperties::toArray () const [pure virtual]`

Method that serializes the contents of the property map to an array.

Returns

list of pairs where the first is the name and the second is the value.

Implemented in `activemq::util::ActiveMQProperties` (p. 378).

6.175.3.11 `virtual std::string cms::CMSProperties::toString () const [pure virtual]`

Formats the contents of the Properties Object into a string that can be logged, etc.

Returns

string value of this object.

Implemented in `activemq::util::ActiveMQProperties` (p. 379).

The documentation for this class was generated from the following file:

- `src/main/cms/CMSProperties.h`

6.176 cms::CMSSecurityException Class Reference

This exception must be thrown when a provider rejects a user name/password submitted by a client.

```
#include <src/main/cms/CMSSecurityException.h>
```

Inheritance diagram for cms::CMSSecurityException:

Public Member Functions

- **CMSSecurityException** () throw ()
- **CMSSecurityException** (const **CMSSecurityException** &ex) throw ()
- **CMSSecurityException** (const std::string &message, const std::exception *cause) throw ()
- **CMSSecurityException** (const std::string &message, const std::exception *cause, const std::vector< std::pair< std::string, int > > &stackTrace) throw ()
- virtual ~**CMSSecurityException** () throw ()

6.176.1 Detailed Description

This exception must be thrown when a provider rejects a user name/password submitted by a client. It may also be thrown for any case where a security restriction prevents a method from completing.

Since

1.3

6.176.2 Constructor & Destructor Documentation

- 6.176.2.1** `cms::CMSSecurityException::CMSSecurityException () throw ()`
- 6.176.2.2** `cms::CMSSecurityException::CMSSecurityException (const CMSSecurityException & ex) throw ()`
- 6.176.2.3** `cms::CMSSecurityException::CMSSecurityException (const std::string & message, const std::exception * cause) throw ()`
- 6.176.2.4** `cms::CMSSecurityException::CMSSecurityException (const std::string & message, const std::exception * cause, const std::vector< std::pair< std::string, int > > & stackTrace) throw ()`
- 6.176.2.5** `virtual cms::CMSSecurityException::~~CMSSecurityException () throw () [virtual]`

The documentation for this class was generated from the following file:

- `src/main/cms/CMSSecurityException.h`

6.177 activemq::cmsutil::CmsTemplate Class Reference

CmsTemplate (p. 969) simplifies performing synchronous CMS operations.

```
#include <src/main/activemq/cmsutil/CmsTemplate.h>
```

Inheritance diagram for `activemq::cmsutil::CmsTemplate`:

Data Structures

- class **ProducerExecutor**
- class **ReceiveExecutor**
- class **ResolveProducerExecutor**
- class **ResolveReceiveExecutor**
- class **SendExecutor**

Public Member Functions

- **CmsTemplate** ()
- **CmsTemplate** (cms::ConnectionFactory *connectionFactory)
- virtual **~CmsTemplate** ()
- virtual void **setDefaultDestination** (cms::Destination *defaultDestination)
Sets the destination object to be used by default for send/receive operations.
- virtual const cms::Destination * **getDefaultDestination** () const
Retrieves the default destination to be used for send/receive operations.
- virtual cms::Destination * **getDefaultDestination** ()
Retrieves the default destination to be used for send/receive operations.
- virtual void **setDefaultDestinationName** (const std::string &defaultDestinationName)
Sets the name of the default destination to be used from send/receive operations.
- virtual const std::string **getDefaultDestinationName** () const
Gets the name of the default destination to be used for send/receive operations.
- virtual void **setPubSubDomain** (bool pubSubDomain)
Indicates whether the default destination is a topic (true) or a queue (false).
- virtual void **setMessageIdEnabled** (bool messageIdEnabled)
- virtual bool **isMessageIdEnabled** () const
- virtual void **setMessageTimestampEnabled** (bool messageTimestampEnabled)
- virtual bool **isMessageTimestampEnabled** () const
- virtual void **setNoLocal** (bool noLocal)
- virtual bool **isNoLocal** () const
- virtual void **setReceiveTimeout** (long long receiveTimeout)
- virtual long long **getReceiveTimeout** () const
- virtual void **setExplicitQosEnabled** (bool explicitQosEnabled)
Set if the QOS values (deliveryMode, priority, timeToLive) should be used for sending a message.
- virtual bool **isExplicitQosEnabled** () const
If "true", then the values of deliveryMode, priority, and timeToLive will be used when sending a message.
- virtual void **setDeliveryPersistent** (bool deliveryPersistent)
Set whether message delivery should be persistent or non-persistent, specified as boolean value ("true" or "false").

- virtual void **setDeliveryMode** (int deliveryMode)
Set the delivery mode to use when sending a message.
- virtual int **getDeliveryMode** () const
Return the delivery mode to use when sending a message.
- virtual void **setPriority** (int priority)
Set the priority of a message when sending.
- virtual int **getPriority** () const
Return the priority of a message when sending.
- virtual void **setTimeToLive** (long long timeToLive)
Set the time-to-live of the message when sending.
- virtual long long **getTimeToLive** () const
Return the time-to-live of the message when sending.
- virtual void **execute** (**SessionCallback** *action) throw (cms::CMSEException)
Executes the given action within a CMS Session.
- virtual void **execute** (**ProducerCallback** *action) throw (cms::CMSEException)
Executes the given action and provides it with a CMS Session and producer.
- virtual void **execute** (cms::Destination *dest, **ProducerCallback** *action) throw (cms::CMSEException)
Executes the given action and provides it with a CMS Session and producer.
- virtual void **execute** (const std::string &destinationName, **ProducerCallback** *action) throw (cms::CMSEException)
Executes the given action and provides it with a CMS Session and producer.
- virtual void **send** (**MessageCreator** *messageCreator) throw (cms::CMSEException)
Convenience method for sending a message to the default destination.
- virtual void **send** (cms::Destination *dest, **MessageCreator** *messageCreator) throw (cms::CMSEException)
Convenience method for sending a message to the specified destination.
- virtual void **send** (const std::string &destinationName, **MessageCreator** *messageCreator) throw (cms::CMSEException)
Convenience method for sending a message to the specified destination.
- virtual cms::Message * **receive** () throw (cms::CMSEException)
Performs a synchronous read from the default destination.
- virtual cms::Message * **receive** (cms::Destination *destination) throw (cms::CMSEException)
Performs a synchronous read from the specified destination.

- virtual **cms::Message** * **receive** (const std::string &destinationName) throw (cms::CMSEException)
Performs a synchronous read from the specified destination.
- virtual **cms::Message** * **receiveSelected** (const std::string &selector) throw (cms::CMSEException)
Performs a synchronous read consuming only messages identified by the given selector.
- virtual **cms::Message** * **receiveSelected** (cms::Destination *destination, const std::string &selector) throw (cms::CMSEException)
Performs a synchronous read from the specified destination, consuming only messages identified by the given selector.
- virtual **cms::Message** * **receiveSelected** (const std::string &destinationName, const std::string &selector) throw (cms::CMSEException)
Performs a synchronous read from the specified destination, consuming only messages identified by the given selector.

Static Public Attributes

- static const long long **RECEIVE_TIMEOUT_NO_WAIT** = -1
Timeout value indicating that a receive operation should check if a message is immediately available without blocking.
- static const long long **RECEIVE_TIMEOUT_INDEFINITE_WAIT** = 0
Timeout value indicating a blocking receive without timeout.
- static const int **DEFAULT_PRIORITY** = 4
Default message priority.
- static const long long **DEFAULT_TIME_TO_LIVE** = 0
My default, messages should live forever.

Protected Member Functions

- void **init** () throw (cms::CMSEException, decaf::lang::exceptions::IllegalStateException)
Initializes this object and prepares it for use.
- void **destroy** () throw (cms::CMSEException, decaf::lang::exceptions::IllegalStateException)
Clears all internal resources.

Friends

- class **ProducerExecutor**
- class **ResolveProducerExecutor**
- class **SendExecutor**

- class **ReceiveExecutor**
- class **ResolveReceiveExecutor**

6.177.1 Detailed Description

CmsTemplate (p. 969) simplifies performing synchronous CMS operations. This class is intended to be for CMS what Spring's **JmsTemplate** is for JMS. Provided with a CMS **ConnectionFactory**, creates and manages all other CMS resources internally.

Before using **CmsTemplate** (p. 969) the user must first set the destination (either by name or by setting the destination object directly) and then call **init** to initialize the object for use.

CmsTemplate (p. 969) allows the user to get access to a CMS **Session** through a user-defined **SessionCallback** (p. 2676). Similarly, if the user wants direct access to a CMS **MessageProducer**, it can provide a **ProducerCallback** (p. 2443). As a convenience, the user can bypass having to provide callbacks altogether for sending messages, by calling one of the **send** methods.

See also

SessionCallback (p. 2676)
ProducerCallback (p. 2443)
MessageCreator (p. 2083)

6.177.2 Constructor & Destructor Documentation

6.177.2.1 `activemq::cmsutil::CmsTemplate::CmsTemplate ()`

6.177.2.2 `activemq::cmsutil::CmsTemplate::CmsTemplate (cms::ConnectionFactory * connectionFactory)`

6.177.2.3 `virtual activemq::cmsutil::CmsTemplate::~~CmsTemplate () [virtual]`

6.177.3 Member Function Documentation

6.177.3.1 `void activemq::cmsutil::CmsTemplate::destroy () throw (cms::CMSException, decaf::lang::exceptions::IllegalStateException) [protected, virtual]`

Clears all internal resources.

Reimplemented from **activemq::cmsutil::CmsDestinationAccessor** (p. 959).

6.177.3.2 `virtual void activemq::cmsutil::CmsTemplate::execute (SessionCallback * action) throw (cms::CMSException) [virtual]`

Executes the given action within a CMS Session.

Parameters

action the action to perform within a CMS Session

Exceptions

cms::CMSException (p. 960) thrown if an error occurs.

6.177.3.3 virtual void activemq::cmsutil::CmsTemplate::execute (ProducerCallback * *action*) throw (cms::CMSException) [virtual]

Executes the given action and provides it with a CMS Session and producer.

Parameters

action the action to perform

Exceptions

cms::CMSException (p. 960) thrown if an error occurs.

6.177.3.4 virtual void activemq::cmsutil::CmsTemplate::execute (cms::Destination * *dest*, ProducerCallback * *action*) throw (cms::CMSException) [virtual]

Executes the given action and provides it with a CMS Session and producer.

Parameters

dest the destination to send messages to

action the action to perform

Exceptions

cms::CMSException (p. 960) thrown if an error occurs.

6.177.3.5 virtual void activemq::cmsutil::CmsTemplate::execute (const std::string & *destinationName*, ProducerCallback * *action*) throw (cms::CMSException) [virtual]

Executes the given action and provides it with a CMS Session and producer.

Parameters

destinationName the name of the destination to send messages to (to internally be resolved to an actual destination)

action the action to perform

Exceptions

cms::CMSException (p. 960) thrown if an error occurs.

6.177.3.6 virtual cms::Destination* activemq::cmsutil::CmsTemplate::getDefaultDestination () [inline, virtual]

Retrieves the default destination to be used for send/receive operations.

Returns

the default destination. Non-const version of this method.

6.177.3.7 `virtual const cms::Destination* activemq::cmsutil::CmsTemplate::getDefaultDestination () const [inline, virtual]`

Retrieves the default destination to be used for send/receive operations.

Returns

the default destination. Const version of this method.

6.177.3.8 `virtual const std::string activemq::cmsutil::CmsTemplate::getDefaultDestinationName () const [inline, virtual]`

Gets the name of the default destination to be used for send/receive operations.

The destination type (topic/queue) is determined by the `pubSubDomain` property.

Returns

the default name of the destination for send/receive operations.

6.177.3.9 `virtual int activemq::cmsutil::CmsTemplate::getDeliveryMode () const [inline, virtual]`

Return the delivery mode to use when sending a message.

6.177.3.10 `virtual int activemq::cmsutil::CmsTemplate::getPriority () const [inline, virtual]`

Return the priority of a message when sending.

6.177.3.11 `virtual long long activemq::cmsutil::CmsTemplate::getReceiveTimeout () const [inline, virtual]`

6.177.3.12 `virtual long long activemq::cmsutil::CmsTemplate::getTimeToLive () const [inline, virtual]`

Return the time-to-live of the message when sending.

6.177.3.13 `void activemq::cmsutil::CmsTemplate::init () throw (cms::CMSException, decaf::lang::exceptions::IllegalStateException) [protected, virtual]`

Initializes this object and prepares it for use.

This should be called before any other methods are called.

Reimplemented from `activemq::cmsutil::CmsDestinationAccessor` (p. 959).

6.177.3.14 `virtual bool activemq::cmsutil::CmsTemplate::isExplicitQosEnabled () const [inline, virtual]`

If "true", then the values of deliveryMode, priority, and timeToLive will be used when sending a message.

Otherwise, the default values, that may be set administratively, will be used.

Returns

true if overriding default values of QOS parameters (deliveryMode, priority, and timeToLive)

See also

`setDeliveryMode` (p. 979)

`setPriority` (p. 980)

`setTimeToLive` (p. 981)

6.177.3.15 `virtual bool activemq::cmsutil::CmsTemplate::isMessageIdEnabled () const [inline, virtual]`

6.177.3.16 `virtual bool activemq::cmsutil::CmsTemplate::isMessageTimestampEnabled () const [inline, virtual]`

6.177.3.17 `virtual bool activemq::cmsutil::CmsTemplate::isNoLocal () const [inline, virtual]`

6.177.3.18 `virtual cms::Message* activemq::cmsutil::CmsTemplate::receive (cms::Destination * destination) throw (cms::CMSException) [virtual]`

Performs a synchronous read from the specified destination.

Parameters

destination the destination to receive on

Returns

the message

Exceptions

cms::CMSException (p. 960) thrown if an error occurs

6.177.3.19 `virtual cms::Message* activemq::cmsutil::CmsTemplate::receive (const std::string & destinationName) throw (cms::CMSException) [virtual]`

Performs a synchronous read from the specified destination.

Parameters

destinationName the name of the destination to receive on (will be resolved to destination internally).

Returns

the message

Exceptions

cms::CMSEException (p. 960) thrown if an error occurs

6.177.3.20 `virtual cms::Message* activemq::cmsutil::CmsTemplate::receive ()
 throw (cms::CMSEException) [virtual]`

Performs a synchronous read from the default destination.

Returns

the message

Exceptions

cms::CMSEException (p. 960) thrown if an error occurs

6.177.3.21 `virtual cms::Message* activemq::cmsutil::CmsTemplate::receiveSelected
 (const std::string & selector) throw (cms::CMSEException)
 [virtual]`

Performs a synchronous read consuming only messages identified by the given selector.

Parameters

selector the selector expression.

Returns

the message

Exceptions

cms::CMSEException (p. 960) thrown if an error occurs

6.177.3.22 `virtual cms::Message* activemq::cmsutil::CmsTemplate::receiveSelected
 (cms::Destination * destination, const std::string & selector) throw
 (cms::CMSEException) [virtual]`

Performs a synchronous read from the specified destination, consuming only messages identified by the given selector.

Parameters

destination the destination to receive on.

selector the selector expression.

Returns

the message

Exceptions

cms::CMSException (p. 960) thrown if an error occurs

6.177.3.23 `virtual cms::Message* activemq::cmsutil::CmsTemplate::receiveSelected (const std::string & destinationName, const std::string & selector) throw (cms::CMSException)` [virtual]

Performs a synchronous read from the specified destination, consuming only messages identified by the given selector.

Parameters

destinationName the name of the destination to receive on (will be resolved to destination internally).

selector the selector expression.

Returns

the message

Exceptions

cms::CMSException (p. 960) thrown if an error occurs

6.177.3.24 `virtual void activemq::cmsutil::CmsTemplate::send (cms::Destination * dest, MessageCreator * messageCreator) throw (cms::CMSException)` [virtual]

Convenience method for sending a message to the specified destination.

Parameters

dest The destination to send to

messageCreator Responsible for creating the message to be sent

Exceptions

cms::CMSException (p. 960) thrown if an error occurs.

6.177.3.25 `virtual void activemq::cmsutil::CmsTemplate::send (const std::string & destinationName, MessageCreator * messageCreator) throw (cms::CMSException)` [virtual]

Convenience method for sending a message to the specified destination.

Parameters

destinationName The name of the destination to send to.

messageCreator Responsible for creating the message to be sent

Exceptions

cms::CMSException (p. 960) thrown if an error occurs.

6.177.3.26 `virtual void activemq::cmsutil::CmsTemplate::send (MessageCreator * messageCreator) throw (cms::CMSException) [virtual]`

Convenience method for sending a message to the default destination.

Parameters

messageCreator Responsible for creating the message to be sent

Exceptions

cms::CMSException (p. 960) thrown if an error occurs.

6.177.3.27 `virtual void activemq::cmsutil::CmsTemplate::setDefaultDestination (cms::Destination * defaultDestination) [inline, virtual]`

Sets the destination object to be used by default for send/receive operations.

If no default destination is provided, the `defaultDestinationName` property is used to resolve this default destination for send/receive operations.

Parameters

defaultDestination the default destination

6.177.3.28 `virtual void activemq::cmsutil::CmsTemplate::setDefaultDestinationName (const std::string & defaultDestinationName) [inline, virtual]`

Sets the name of the default destination to be used from send/receive operations.

Calling this method will set the `defaultDestination` property to NULL. The destination type (topic/queue) is determined by the `pubSubDomain` property.

Parameters

defaultDestinationName the name of the destination for send/receive to by default.

6.177.3.29 `virtual void activemq::cmsutil::CmsTemplate::setDeliveryMode (int deliveryMode) [inline, virtual]`

Set the delivery mode to use when sending a message.

Default is the Message default: "PERSISTENT".

Since a default value may be defined administratively, this is only used when "isExplicitQosEnabled" equals "true".

Parameters

deliveryMode the delivery mode to use

See also

`isExplicitQosEnabled` (p. 976)

6.177.3.30 virtual void activemq::cmsutil::CmsTemplate::setDeliveryPersistent (bool *deliveryPersistent*) [inline, virtual]

Set whether message delivery should be persistent or non-persistent, specified as boolean value ("true" or "false").

This will set the delivery mode accordingly, to either "PERSISTENT" or "NON_PERSISTENT". Default it "true" aka delivery mode "PERSISTENT".

See also

`setDeliveryMode(int)` (p. 979)

6.177.3.31 virtual void activemq::cmsutil::CmsTemplate::setExplicitQosEnabled (bool *explicitQosEnabled*) [inline, virtual]

Set if the QOS values (deliveryMode, priority, timeToLive) should be used for sending a message.

See also

`setDeliveryMode` (p. 979)

`setPriority` (p. 980)

`setTimeToLive` (p. 981)

6.177.3.32 virtual void activemq::cmsutil::CmsTemplate::setMessageIdEnabled (bool *messageIdEnabled*) [inline, virtual]**6.177.3.33 virtual void activemq::cmsutil::CmsTemplate::setMessageTimestampEnabled (bool *messageTimestampEnabled*) [inline, virtual]****6.177.3.34 virtual void activemq::cmsutil::CmsTemplate::setNoLocal (bool *noLocal*) [inline, virtual]****6.177.3.35 virtual void activemq::cmsutil::CmsTemplate::setPriority (int *priority*) [inline, virtual]**

Set the priority of a message when sending.

Since a default value may be defined administratively, this is only used when "isExplicitQosEnabled" equals "true".

See also

`isExplicitQosEnabled` (p. 976)

6.177.3.36 virtual void activemq::cmsutil::CmsTemplate::setPubSubDomain (bool *pubSubDomain*) [inline, virtual]

Indicates whether the default destination is a topic (true) or a queue (false).

Calling this method will set the `defaultDestination` property to NULL.

Parameters

pubSubDomain indicates whether to use pub-sub messaging (topics).

Reimplemented from **activemq::cmsutil::CmsDestinationAccessor** (p. 960).

References **activemq::cmsutil::CmsDestinationAccessor::setPubSubDomain()**.

6.177.3.37 `virtual void activemq::cmsutil::CmsTemplate::setReceiveTimeout (long long receiveTimeout)` [inline, virtual]

6.177.3.38 `virtual void activemq::cmsutil::CmsTemplate::setTimeToLive (long long timeToLive)` [inline, virtual]

Set the time-to-live of the message when sending.

Since a default value may be defined administratively, this is only used when "isExplicitQosEnabled" equals "true".

Parameters

timeToLive the message's lifetime (in milliseconds)

See also

isExplicitQosEnabled (p. 976)

6.177.4 Friends And Related Function Documentation

6.177.4.1 `friend class ProducerExecutor` [friend]

6.177.4.2 `friend class ReceiveExecutor` [friend]

6.177.4.3 `friend class ResolveProducerExecutor` [friend]

6.177.4.4 `friend class ResolveReceiveExecutor` [friend]

6.177.4.5 `friend class SendExecutor` [friend]

6.177.5 Field Documentation

6.177.5.1 `const int activemq::cmsutil::CmsTemplate::DEFAULT_PRIORITY = 4` [static]

Default message priority.

6.177.5.2 `const long long activemq::cmsutil::CmsTemplate::DEFAULT_TIME_TO_LIVE = 0` [static]

My default, messages should live forever.

6.177.5.3 `const long long activemq::cmsutil::CmsTemplate::RECEIVE _ - TIMEOUT _ INDEFINITE _ WAIT = 0 [static]`

Timeout value indicating a blocking receive without timeout.

6.177.5.4 `const long long activemq::cmsutil::CmsTemplate::RECEIVE _ - TIMEOUT _ NO _ WAIT = -1 [static]`

Timeout value indicating that a receive operation should check if a message is immediately available without blocking.

The documentation for this class was generated from the following file:

- `src/main/activemq/cmsutil/CmsTemplate.h`

6.178 `decaf::util::Collection< E >` Class Template Reference

The root interface in the collection hierarchy.

```
#include <src/main/decaf/util/Collection.h>
```

Inheritance diagram for `decaf::util::Collection< E >`:

Public Member Functions

- virtual `~Collection ()`
- virtual `bool add (const E &value)=0 throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException)`
Returns true if this collection changed as a result of the call.
- virtual `bool addAll (const Collection< E > &collection)=0 throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException)`
Adds all of the elements in the specified collection to this collection.
- virtual `void clear ()=0 throw (lang::exceptions::UnsupportedOperationException)`
Removes all of the elements from this collection (optional operation).
- virtual `bool contains (const E &value) const =0 throw (lang::Exception)`
Returns true if this collection contains the specified element.
- virtual `bool containsAll (const Collection< E > &collection) const =0 throw (lang::Exception)`
Returns true if this collection contains all of the elements in the specified collection.
- virtual `bool equals (const Collection< E > &value) const =0`
Compares the passed collection to this one, if they contain the same elements, i.e.
- virtual `bool isEmpty () const =0`

- virtual bool **remove** (const E &value)=0 throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException)
Removes a single instance of the specified element from the collection.
- virtual bool **removeAll** (const **Collection**< E > &collection)=0 throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException)
Removes all this collection's elements that are also contained in the specified collection (optional operation).
- virtual bool **retainAll** (const **Collection**< E > &collection)=0 throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException)
Retains only the elements in this collection that are contained in the specified collection (optional operation).
- virtual std::size_t **size** () const =0
Returns the number of elements in this collection.
- virtual std::vector< E > **toArray** () const =0
Returns an array containing all of the elements in this collection.

6.178.1 Detailed Description

template<typename E> class decaf::util::Collection< E >

The root interface in the collection hierarchy. A collection represents a group of objects, known as its elements. Some collections allow duplicate elements and others do not. Some are ordered and others unordered. This interface is typically used to pass collections around and manipulate them where maximum generality is desired.

All general-purpose **Collection** (p. 982) implementation classes (which typically implement **Collection** (p. 982) indirectly through one of its subinterfaces) should provide two "standard" constructors: a void (no arguments) constructor, which creates an empty collection, and a constructor with a single argument of type **Collection** (p. 982), which creates a new collection with the same elements as its argument. In effect, the latter constructor allows the user to copy any collection, producing an equivalent collection of the desired implementation type. There is no way to enforce this convention (as interfaces cannot contain constructors) but all of the general-purpose **Collection** (p. 982) implementations in the Decaf platform libraries comply.

The "destructive" methods contained in this interface, that is, the methods that modify the collection on which they operate, are specified to throw `UnsupportedOperationException` if this collection does not support the operation. If this is the case, these methods may, but are not required to, throw an `UnsupportedOperationException` if the invocation would have no effect on the collection. For example, invoking the `addAll(Collection)` method on an unmodifiable collection may, but is not required to, throw the exception if the collection to be added is empty.

Many methods in Collections Framework interfaces are defined in terms of the `equals` method. For example, the specification for the `contains(Object o)` method says: "returns true if and only if this collection contains at least one element `e` such that `(o==null ? e==null : o.equals(e))`."

Since

1.0

6.178.2 Constructor & Destructor Documentation

6.178.2.1 `template<typename E> virtual decaf::util::Collection< E >::~~Collection () [inline, virtual]`

6.178.3 Member Function Documentation

6.178.3.1 `template<typename E> virtual bool decaf::util::Collection< E >::add (const E & value) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException) [pure virtual]`

Returns true if this collection changed as a result of the call.

(Returns false if this collection does not permit duplicates and already contains the specified element.)

Collections that support this operation may place limitations on what elements may be added to this collection. In particular, some collections will refuse to add null elements, and others will impose restrictions on the type of elements that may be added. **Collection** (p. 982) classes should clearly specify in their documentation any restrictions on what elements may be added.

If a collection refuses to add a particular element for any reason other than that it already contains the element, it must throw an exception (rather than returning false). This preserves the invariant that a collection always contains the specified element after this call returns.

For non-pointer values, i.e. class instances or string's the object will be copied into the collection, thus the object must support being copied, must not hide the copy constructor and assignment operator.

Parameters

value - reference to the element to add.

Returns

true if the element was added

Exceptions

UnsupportedOperationException

IllegalArgumentException

IllegalStateException if the element cannot be added at this time due to insertion restrictions

Implemented in `decaf::util::AbstractQueue< E >` (p. 135), `decaf::util::PriorityQueue< E >` (p. 2416), `decaf::util::StlList< E >` (p. 2838), `decaf::util::StlSet< E >` (p. 2868), `decaf::util::StlList< CompositeTask * >` (p. 2838), `decaf::util::StlList< URI >` (p. 2838), `decaf::util::StlList< Pointer< DestinationInfo > >` (p. 2838), `decaf::util::StlList< PrimitiveValueNode >` (p. 2838), `decaf::util::StlList< Pointer< Command > >` (p. 2838), `decaf::util::StlList< Pointer< BackupTransport > >` (p. 2838), `decaf::util::StlSet< transport::TransportListener * >` (p. 2868), `decaf::util::StlSet< Pointer< Synchronization > >` (p. 2868), and `decaf::util::StlSet< ActiveMQSession * >` (p. 2868).

```

6.178.3.2  template<typename E> virtual bool decaf::util::Collection<
             E >::addAll ( const Collection< E > & collection ) throw
             ( lang::exceptions::UnsupportedOperationException,
             lang::exceptions::IllegalArgumentException,
             lang::exceptions::IllegalStateException ) [pure
             virtual]

```

Adds all of the elements in the specified collection to this collection.

The behavior of this operation is undefined if the specified collection is modified while the operation is in progress. (This implies that the behavior of this call is undefined if the specified collection is this collection, and this collection is nonempty.)

Parameters

collection - **Collection** (p. 982) whose elements are added to this one.

Returns

true if this collection changed as a result of the call

Exceptions

UnsupportedOperationException

IllegalArgumentException

IllegalStateException if the element cannot be added at this time due to insertion restrictions

Implemented in **decaf::util::AbstractCollection< E >** (p. 122), **decaf::util::AbstractQueue< E >** (p. 135), **decaf::util::AbstractCollection< transport::TransportListener * >** (p. 122), **decaf::util::AbstractCollection< Pointer< Synchronization > >** (p. 122), **decaf::util::AbstractCollection< CompositeTask * >** (p. 122), **decaf::util::AbstractCollection< URI >** (p. 122), **decaf::util::AbstractCollection< ActiveMQSession * >** (p. 122), **decaf::util::AbstractCollection< Pointer< DestinationInfo > >** (p. 122), **decaf::util::AbstractCollection< PrimitiveValueNode >** (p. 122), **decaf::util::AbstractCollection< Pointer< Command > >** (p. 122), and **decaf::util::AbstractCollection< Pointer< BackupTransport > >** (p. 122).

```

6.178.3.3  template<typename E> virtual void decaf::util::Collection< E >::clear (
             ) throw ( lang::exceptions::UnsupportedOperationException ) [pure
             virtual]

```

Removes all of the elements from this collection (optional operation).

This collection will be empty after this method returns unless it throws an exception.

Exceptions

UnsupportedOperationException

Implemented in **decaf::util::AbstractCollection< E >** (p. 123), **decaf::util::AbstractQueue< E >** (p. 136), **decaf::util::PriorityQueue< E >** (p. 2417), **decaf::util::StlList< E >** (p. 2840), **decaf::util::StlSet< E >** (p. 2869), **decaf::util::AbstractCollection< transport::TransportListener * >** (p. 123), **decaf::util::AbstractCollection< Pointer< Synchronization > >**

> (p. 123), decaf::util::AbstractCollection< CompositeTask * > (p. 123), decaf::util::AbstractCollection< URI > (p. 123), decaf::util::AbstractCollection< ActiveMQSession * > (p. 123), decaf::util::AbstractCollection< Pointer< DestinationInfo > > (p. 123), decaf::util::AbstractCollection< PrimitiveValueNode > (p. 123), decaf::util::AbstractCollection< Pointer< Command > > (p. 123), decaf::util::AbstractCollection< Pointer< BackupTransport > > (p. 123), decaf::util::StlList< CompositeTask * > (p. 2840), decaf::util::StlList< URI > (p. 2840), decaf::util::StlList< Pointer< DestinationInfo > > (p. 2840), decaf::util::StlList< PrimitiveValueNode > (p. 2840), decaf::util::StlList< Pointer< Command > > (p. 2840), decaf::util::StlList< Pointer< BackupTransport > > (p. 2840), decaf::util::StlSet< transport::TransportListener * > (p. 2869), decaf::util::StlSet< Pointer< Synchronization > > (p. 2869), and decaf::util::StlSet< ActiveMQSession * > (p. 2869).

6.178.3.4 `template<typename E> virtual bool decaf::util::Collection< E >::contains (const E & value) const throw (lang::Exception) [pure virtual]`

Returns true if this collection contains the specified element.

More formally, returns true if and only if this collection contains at least one element *e* such that (*o*==null ? *e*==null : *o.equals(e)*).

Parameters

value - value to check for presence in the collection

Returns

true if there is at least one of the elements in the collection

Exceptions

Exception

Implemented in decaf::util::AbstractCollection< E > (p. 124), decaf::util::StlList< E > (p. 2840), decaf::util::StlSet< E > (p. 2869), decaf::util::AbstractCollection< transport::TransportListener * > (p. 124), decaf::util::AbstractCollection< Pointer< Synchronization > > (p. 124), decaf::util::AbstractCollection< CompositeTask * > (p. 124), decaf::util::AbstractCollection< URI > (p. 124), decaf::util::AbstractCollection< ActiveMQSession * > (p. 124), decaf::util::AbstractCollection< Pointer< DestinationInfo > > (p. 124), decaf::util::AbstractCollection< PrimitiveValueNode > (p. 124), decaf::util::AbstractCollection< Pointer< Command > > (p. 124), decaf::util::AbstractCollection< Pointer< BackupTransport > > (p. 124), decaf::util::StlList< CompositeTask * > (p. 2840), decaf::util::StlList< URI > (p. 2840), decaf::util::StlList< Pointer< DestinationInfo > > (p. 2840), decaf::util::StlList< PrimitiveValueNode > (p. 2840), decaf::util::StlList< Pointer< Command > > (p. 2840), decaf::util::StlList< Pointer< BackupTransport > > (p. 2840), decaf::util::StlSet< transport::TransportListener * > (p. 2869), decaf::util::StlSet< Pointer< Synchronization > > (p. 2869), and decaf::util::StlSet< ActiveMQSession * > (p. 2869).

6.178.3.5 `template<typename E> virtual bool decaf::util::Collection< E >::containsAll (const Collection< E > & collection) const throw (lang::Exception) [pure virtual]`

Returns true if this collection contains all of the elements in the specified collection.

Parameters

collection - **Collection** (p. 982) to compare to this one.

Exceptions

Exception

Implemented in **decaf::util::AbstractCollection< E >** (p. 124), **decaf::util::AbstractCollection< transport::TransportListener * >** (p. 124), **decaf::util::AbstractCollection< Pointer< Synchronization > >** (p. 124), **decaf::util::AbstractCollection< CompositeTask * >** (p. 124), **decaf::util::AbstractCollection< URI >** (p. 124), **decaf::util::AbstractCollection< ActiveMQSession * >** (p. 124), **decaf::util::AbstractCollection< Pointer< DestinationInfo > >** (p. 124), **decaf::util::AbstractCollection< PrimitiveValueNode >** (p. 124), **decaf::util::AbstractCollection< Pointer< Command > >** (p. 124), and **decaf::util::AbstractCollection< Pointer< BackupTransport > >** (p. 124).

6.178.3.6 `template<typename E> virtual bool decaf::util::Collection< E >::equals (const Collection< E > & value) const` [pure virtual]

Compares the passed collection to this one, if they contain the same elements, i.e. all their elements are equivalent, then it returns true.

Returns

true if the Collections contain the same elements.

Implemented in **decaf::util::AbstractCollection< E >** (p. 125), **decaf::util::AbstractCollection< transport::TransportListener * >** (p. 125), **decaf::util::AbstractCollection< Pointer< Synchronization > >** (p. 125), **decaf::util::AbstractCollection< CompositeTask * >** (p. 125), **decaf::util::AbstractCollection< URI >** (p. 125), **decaf::util::AbstractCollection< ActiveMQSession * >** (p. 125), **decaf::util::AbstractCollection< Pointer< DestinationInfo > >** (p. 125), **decaf::util::AbstractCollection< PrimitiveValueNode >** (p. 125), **decaf::util::AbstractCollection< Pointer< Command > >** (p. 125), and **decaf::util::AbstractCollection< Pointer< BackupTransport > >** (p. 125).

6.178.3.7 `template<typename E> virtual bool decaf::util::Collection< E >::isEmpty () const` [pure virtual]

Returns

true if this collection contains no elements.

Implemented in **decaf::util::AbstractCollection< E >** (p. 125), **decaf::util::StlList< E >** (p. 2841), **decaf::util::StlSet< E >** (p. 2870), **decaf::util::AbstractCollection< transport::TransportListener * >** (p. 125), **decaf::util::AbstractCollection< Pointer< Synchronization > >** (p. 125), **decaf::util::AbstractCollection< CompositeTask * >** (p. 125), **decaf::util::AbstractCollection< URI >** (p. 125), **decaf::util::AbstractCollection< ActiveMQSession * >** (p. 125), **decaf::util::AbstractCollection< Pointer< DestinationInfo > >** (p. 125), **decaf::util::AbstractCollection< PrimitiveValueNode >** (p. 125), **decaf::util::AbstractCollection< Pointer< Command > >** (p. 125), and **decaf::util::AbstractCollection< Pointer< BackupTransport > >** (p. 125),

decaf::util::StlList< CompositeTask * > (p. 2841), decaf::util::StlList< URI > (p. 2841), decaf::util::StlList< Pointer< DestinationInfo > > (p. 2841), decaf::util::StlList< PrimitiveValueNode > (p. 2841), decaf::util::StlList< Pointer< Command > > (p. 2841), decaf::util::StlList< Pointer< BackupTransport > > (p. 2841), decaf::util::StlSet< transport::TransportListener * > (p. 2870), decaf::util::StlSet< Pointer< Synchronization > > (p. 2870), and decaf::util::StlSet< ActiveMQSession * > (p. 2870).

6.178.3.8 `template<typename E> virtual bool decaf::util::Collection< E >::remove (const E & value) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException) [pure virtual]`

Removes a single instance of the specified element from the collection.

More formally, removes an element *e* such that (*o*==null ? *e*==null : *o*.equals(*e*)), if this collection contains one or more such elements. Returns true if this collection contained the specified element (or equivalently, if this collection changed as a result of the call).

Parameters

value - reference to the element to remove.

Returns

true if the collection was changed

Exceptions

UnsupportedOperationException

IllegalArgumentException

Implemented in decaf::util::AbstractCollection< E > (p. 127), decaf::util::PriorityQueue< E > (p. 2419), decaf::util::StlList< E > (p. 2843), decaf::util::StlSet< E > (p. 2870), decaf::util::AbstractCollection< transport::TransportListener * > (p. 127), decaf::util::AbstractCollection< Pointer< Synchronization > > (p. 127), decaf::util::AbstractCollection< CompositeTask * > (p. 127), decaf::util::AbstractCollection< URI > (p. 127), decaf::util::AbstractCollection< ActiveMQSession * > (p. 127), decaf::util::AbstractCollection< Pointer< DestinationInfo > > (p. 127), decaf::util::AbstractCollection< PrimitiveValueNode > (p. 127), decaf::util::AbstractCollection< Pointer< Command > > (p. 127), decaf::util::AbstractCollection< Pointer< BackupTransport > > (p. 127), decaf::util::StlList< CompositeTask * > (p. 2843), decaf::util::StlList< URI > (p. 2843), decaf::util::StlList< Pointer< DestinationInfo > > (p. 2843), decaf::util::StlList< PrimitiveValueNode > (p. 2843), decaf::util::StlList< Pointer< Command > > (p. 2843), decaf::util::StlList< Pointer< BackupTransport > > (p. 2843), decaf::util::StlSet< transport::TransportListener * > (p. 2870), decaf::util::StlSet< Pointer< Synchronization > > (p. 2870), and decaf::util::StlSet< ActiveMQSession * > (p. 2870).

6.178.3.9 `template<typename E> virtual bool decaf::util::Collection< E >::removeAll (const Collection< E > & collection) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException) [pure virtual]`

Removes all this collection's elements that are also contained in the specified collection (optional operation).

After this call returns, this collection will contain no elements in common with the specified collection.

Parameters

collection - The **Collection** (p. 982) whose elements are to be removed

Returns

true if the collection changed as a result of this call

Exceptions

UnsupportedOperationException

IllegalArgumentException

Implemented in `decaf::util::AbstractCollection< E >` (p. 128), `decaf::util::AbstractSet< E >` (p. 139), `decaf::util::AbstractCollection< transport::TransportListener * >` (p. 128), `decaf::util::AbstractCollection< Pointer< Synchronization > >` (p. 128), `decaf::util::AbstractCollection< CompositeTask * >` (p. 128), `decaf::util::AbstractCollection< URI >` (p. 128), `decaf::util::AbstractCollection< ActiveMQSession * >` (p. 128), `decaf::util::AbstractCollection< Pointer< DestinationInfo > >` (p. 128), `decaf::util::AbstractCollection< Primitive-ValueNode >` (p. 128), `decaf::util::AbstractCollection< Pointer< Command > >` (p. 128), `decaf::util::AbstractCollection< Pointer< BackupTransport > >` (p. 128), `decaf::util::AbstractSet< transport::TransportListener * >` (p. 139), `decaf::util::AbstractSet< Pointer< Synchronization > >` (p. 139), and `decaf::util::AbstractSet< ActiveMQSession * >` (p. 139).

```
6.178.3.10  template<typename E> virtual bool decaf::util::Collection<
              E >::retainAll ( const Collection< E > & collection )
              throw ( lang::exceptions::UnsupportedOperationException,
                      lang::exceptions::IllegalArgumentException ) [pure virtual]
```

Retains only the elements in this collection that are contained in the specified collection (optional operation).

In other words, removes from this collection all of its elements that are not contained in the specified collection.

Parameters

collection - The **Collection** (p. 982) whose elements are to be retained

Returns

true if the collection changed as a result of this call

Exceptions

UnsupportedOperationException

IllegalArgumentException

Implemented in `decaf::util::AbstractCollection< E >` (p. 128), `decaf::util::AbstractCollection< transport::TransportListener * >`

(p. 128), `decaf::util::AbstractCollection< Pointer< Synchronization > >` (p. 128), `decaf::util::AbstractCollection< CompositeTask * >` (p. 128), `decaf::util::AbstractCollection< URI >` (p. 128), `decaf::util::AbstractCollection< ActiveMQSession * >` (p. 128), `decaf::util::AbstractCollection< Pointer< DestinationInfo > >` (p. 128), `decaf::util::AbstractCollection< PrimitiveValueNode >` (p. 128), `decaf::util::AbstractCollection< Pointer< Command > >` (p. 128), and `decaf::util::AbstractCollection< Pointer< BackupTransport > >` (p. 128).

6.178.3.11 `template<typename E> virtual std::size_t decaf::util::Collection< E >::size () const [pure virtual]`

Returns the number of elements in this collection.

If this collection contains more than Integer.MAX_VALUE elements, returns Integer.MAX_VALUE.

Returns

the number of elements in this collection

Implemented in `decaf::util::PriorityQueue< E >` (p. 2420), `decaf::util::StlList< E >` (p. 2844), `decaf::util::StlSet< E >` (p. 2871), `decaf::util::StlList< CompositeTask * >` (p. 2844), `decaf::util::StlList< URI >` (p. 2844), `decaf::util::StlList< Pointer< DestinationInfo > >` (p. 2844), `decaf::util::StlList< PrimitiveValueNode >` (p. 2844), `decaf::util::StlList< Pointer< Command > >` (p. 2844), `decaf::util::StlList< Pointer< BackupTransport > >` (p. 2844), `decaf::util::StlSet< transport::TransportListener * >` (p. 2871), `decaf::util::StlSet< Pointer< Synchronization > >` (p. 2871), and `decaf::util::StlSet< ActiveMQSession * >` (p. 2871).

Referenced by `decaf::util::AbstractCollection< Pointer< BackupTransport > >::equals()`, `decaf::util::AbstractCollection< Pointer< BackupTransport > >::isEmpty()`, `decaf::util::AbstractSet< ActiveMQSession * >::removeAll()`, and `decaf::util::AbstractCollection< Pointer< BackupTransport > >::toArray()`.

6.178.3.12 `template<typename E> virtual std::vector<E> decaf::util::Collection< E >::toArray () const [pure virtual]`

Returns an array containing all of the elements in this collection.

If the collection makes any guarantees as to what order its elements are returned by its iterator, this method must return the elements in the same order.

This method acts as bridge between array-based and collection-based APIs.

Returns

an array of the elements in this collection.

Implemented in `decaf::util::AbstractCollection< E >` (p. 129), `decaf::util::AbstractCollection< transport::TransportListener * >` (p. 129), `decaf::util::AbstractCollection< Pointer< Synchronization > >` (p. 129), `decaf::util::AbstractCollection< CompositeTask * >` (p. 129), `decaf::util::AbstractCollection< URI >` (p. 129), `decaf::util::AbstractCollection< ActiveMQSession * >` (p. 129), `decaf::util::AbstractCollection< Pointer< DestinationInfo > >` (p. 129), and `decaf::util::AbstractCollection< PrimitiveValueNode >`

(p. 129), `decaf::util::AbstractCollection< Pointer< Command > >` (p. 129), and `decaf::util::AbstractCollection< Pointer< BackupTransport > >` (p. 129).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/Collection.h`

6.179 activemq::commands::Command Class Reference

```
#include <src/main/activemq/commands/Command.h>
```

Inheritance diagram for `activemq::commands::Command`:

Public Member Functions

- virtual `~Command ()`
- virtual void `setCommandId (int id)=0`
*Sets the **Command** (p. 991) Id of this **Message** (p. 2018).*
- virtual int `getCommandId () const =0`
*Gets the **Command** (p. 991) Id of this **Message** (p. 2018).*
- virtual void `setResponseRequired (const bool required)=0`
*Set if this **Message** (p. 2018) requires a **Response** (p. 2611).*
- virtual bool `isResponseRequired () const =0`
*Is a **Response** (p. 2611) required for this **Command** (p. 991).*
- virtual `std::string toString () const =0`
Returns a provider-specific string that provides information about the contents of the command.
- virtual `decaf::lang::Pointer< commands::Command > visit (activemq::state::CommandVisitor *visitor)=0` throw (exceptions::ActiveMQException)
Allows a Visitor to visit this command and return a response to the command based on the command type being visited.
- virtual bool `isConnectionInfo () const =0`
- virtual bool `isConsumerInfo () const =0`
- virtual bool `isBrokerInfo () const =0`
- virtual bool `isKeepAliveInfo () const =0`
- virtual bool `isMessage () const =0`
- virtual bool `isMessageAck () const =0`
- virtual bool `isMessageDispatch () const =0`
- virtual bool `isMessageDispatchNotification () const =0`
- virtual bool `isProducerAck () const =0`
- virtual bool `isProducerInfo () const =0`
- virtual bool `isResponse () const =0`

- virtual bool **isRemoveInfo** () const =0
- virtual bool **isRemoveSubscriptionInfo** () const =0
- virtual bool **isShutdownInfo** () const =0
- virtual bool **isTransactionInfo** () const =0
- virtual bool **isWireFormatInfo** () const =0

6.179.1 Constructor & Destructor Documentation

6.179.1.1 virtual `activemq::commands::Command::~~Command` () [inline, virtual]

6.179.2 Member Function Documentation

6.179.2.1 virtual `int activemq::commands::Command::getCommandId` () const [pure virtual]

Gets the **Command** (p. 991) Id of this **Message** (p. 2018).

Returns

Command (p. 991) Id

Implemented in `activemq::commands::BaseCommand` (p. 599).

6.179.2.2 virtual bool `activemq::commands::Command::isBrokerInfo` () const [pure virtual]

Implemented in `activemq::commands::BaseCommand` (p. 600), and `activemq::commands::BrokerInfo` (p. 718).

6.179.2.3 virtual bool `activemq::commands::Command::isConnectionInfo` () const [pure virtual]

Implemented in `activemq::commands::BaseCommand` (p. 600), and `activemq::commands::ConnectionInfo` (p. 1132).

6.179.2.4 virtual bool `activemq::commands::Command::isConsumerInfo` () const [pure virtual]

Implemented in `activemq::commands::BaseCommand` (p. 600), and `activemq::commands::ConsumerInfo` (p. 1219).

6.179.2.5 virtual bool `activemq::commands::Command::isKeepAliveInfo` () const [pure virtual]

Implemented in `activemq::commands::BaseCommand` (p. 600), and `activemq::commands::KeepAliveInfo` (p. 1814).

6.179.2.6 `virtual bool activemq::commands::Command::isMessage () const`
[pure virtual]

Implemented in `activemq::commands::BaseCommand` (p. 600), and `activemq::commands::Message` (p. 2030).

6.179.2.7 `virtual bool activemq::commands::Command::isMessageAck () const`
[pure virtual]

Implemented in `activemq::commands::BaseCommand` (p. 600), and `activemq::commands::MessageAck` (p. 2058).

6.179.2.8 `virtual bool activemq::commands::Command::isMessageDispatch () const`
[pure virtual]

Implemented in `activemq::commands::BaseCommand` (p. 600), and `activemq::commands::MessageDispatch` (p. 2087).

6.179.2.9 `virtual bool activemq::commands::Command::isMessageDispatchNotification () const`
[pure virtual]

Implemented in `activemq::commands::BaseCommand` (p. 601), and `activemq::commands::MessageDispatchNotification` (p. 2118).

6.179.2.10 `virtual bool activemq::commands::Command::isProducerAck () const`
[pure virtual]

Implemented in `activemq::commands::BaseCommand` (p. 601), and `activemq::commands::ProducerAck` (p. 2422).

6.179.2.11 `virtual bool activemq::commands::Command::isProducerInfo () const`
[pure virtual]

Implemented in `activemq::commands::BaseCommand` (p. 601), and `activemq::commands::ProducerInfo` (p. 2472).

6.179.2.12 `virtual bool activemq::commands::Command::isRemoveInfo () const`
[pure virtual]

Implemented in `activemq::commands::BaseCommand` (p. 601), and `activemq::commands::RemoveInfo` (p. 2539).

6.179.2.13 `virtual bool activemq::commands::Command::isRemoveSubscriptionInfo () const`
[pure virtual]

Implemented in `activemq::commands::BaseCommand` (p. 601), and `activemq::commands::RemoveSubscriptionInfo` (p. 2563).

6.179.2.14 `virtual bool activemq::commands::Command::isResponse () const`
[pure virtual]

Implemented in `activemq::commands::BaseCommand` (p. 601), and `activemq::commands::Response` (p. 2613).

6.179.2.15 `virtual bool activemq::commands::Command::isResponseRequired () const` [pure virtual]

Is a **Response** (p. 2611) required for this **Command** (p. 991).

Returns

true if a response is required.

Implemented in `activemq::commands::BaseCommand` (p. 601).

6.179.2.16 `virtual bool activemq::commands::Command::isShutdownInfo () const` [pure virtual]

Implemented in `activemq::commands::BaseCommand` (p. 602), and `activemq::commands::ShutdownInfo` (p. 2759).

6.179.2.17 `virtual bool activemq::commands::Command::isTransactionInfo () const` [pure virtual]

Implemented in `activemq::commands::BaseCommand` (p. 602), and `activemq::commands::TransactionInfo` (p. 3039).

6.179.2.18 `virtual bool activemq::commands::Command::isWireFormatInfo () const` [pure virtual]

Implemented in `activemq::commands::BaseCommand` (p. 602), and `activemq::commands::WireFormatInfo` (p. 3159).

6.179.2.19 `virtual void activemq::commands::Command::setCommandId (int id)` [pure virtual]

Sets the **Command** (p. 991) Id of this **Message** (p. 2018).

Parameters

id **Command** (p. 991) Id

Implemented in `activemq::commands::BaseCommand` (p. 602).

6.179.2.20 `virtual void activemq::commands::Command::setResponseRequired (const bool required)` [pure virtual]

Set if this **Message** (p. 2018) requires a **Response** (p. 2611).

Parameters

required true if response is required

Implemented in `activemq::commands::BaseCommand` (p. 602).

6.179.2.21 `virtual std::string activemq::commands::Command::toString () const`
[pure virtual]

Returns a provider-specific string that provides information about the contents of the command.

Reimplemented from `activemq::commands::BaseDataStructure` (p. 662).

Implemented in `activemq::commands::ActiveMQBlobMessage` (p. 146), `activemq::commands::ActiveMQBytesMessage` (p. 177), `activemq::commands::ActiveMQMapMessage` (p. 284), `activemq::commands::ActiveMQMessage` (p. 307), `activemq::commands::ActiveMQObjectMessage` (p. 346), `activemq::commands::ActiveMQStreamMessage` (p. 432), `activemq::commands::ActiveMQTextMessage` (p. 529), `activemq::commands::BaseCommand` (p. 602), `activemq::commands::BrokerInfo` (p. 719), `activemq::commands::ConnectionControl` (p. 1058), `activemq::commands::ConnectionError` (p. 1082), `activemq::commands::ConnectionInfo` (p. 1133), `activemq::commands::ConsumerControl` (p. 1169), `activemq::commands::ConsumerInfo` (p. 1220), `activemq::commands::ControlCommand` (p. 1245), `activemq::commands::DataArrayResponse` (p. 1270), `activemq::commands::DataResponse` (p. 1309), `activemq::commands::DestinationInfo` (p. 1394), `activemq::commands::ExceptionResponse` (p. 1486), `activemq::commands::FlushCommand` (p. 1575), `activemq::commands::IntegerResponse` (p. 1669), `activemq::commands::KeepAliveInfo` (p. 1815), `activemq::commands::Message` (p. 2033), `activemq::commands::MessageAck` (p. 2059), `activemq::commands::MessageDispatch` (p. 2087), `activemq::commands::MessageDispatchNotification` (p. 2119), `activemq::commands::MessagePull` (p. 2206), `activemq::commands::ProducerAck` (p. 2423), `activemq::commands::ProducerInfo` (p. 2473), `activemq::commands::RemoveInfo` (p. 2539), `activemq::commands::RemoveSubscriptionInfo` (p. 2563), `activemq::commands::ReplayCommand` (p. 2586), `activemq::commands::Response` (p. 2614), `activemq::commands::SessionInfo` (p. 2704), `activemq::commands::ShutdownInfo` (p. 2759), `activemq::commands::TransactionInfo` (p. 3040), and `activemq::commands::WireFormatInfo` (p. 3162).

6.179.2.22 `virtual decaf::lang::Pointer<commands::Command> activemq::commands::Command::visit (activemq::state::CommandVisitor * visitor) throw (exceptions::ActiveMQException)` [pure virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

Returns

a **Response** (p. 2611) to the visitor being called or NULL if no response.

Implemented in `activemq::commands::BrokerError` (p. 689), `activemq::commands::BrokerInfo` (p. 720), `activemq::commands::ConnectionControl`

(p. 1059), **activemq::commands::ConnectionError** (p. 1083), **activemq::commands::ConnectionInfo** (p. 1133), **activemq::commands::ConsumerControl** (p. 1170), **activemq::commands::ConsumerInfo** (p. 1221), **activemq::commands::ControlCommand** (p. 1246), **activemq::commands::DestinationInfo** (p. 1394), **activemq::commands::FlushCommand** (p. 1575), **activemq::commands::KeepAliveInfo** (p. 1815), **activemq::commands::Message** (p. 2034), **activemq::commands::MessageAck** (p. 2059), **activemq::commands::MessageDispatch** (p. 2088), **activemq::commands::MessageDispatchNotification** (p. 2119), **activemq::commands::MessagePull** (p. 2206), **activemq::commands::ProducerAck** (p. 2423), **activemq::commands::ProducerInfo** (p. 2473), **activemq::commands::RemoveInfo** (p. 2540), **activemq::commands::RemoveSubscriptionInfo** (p. 2563), **activemq::commands::ReplayCommand** (p. 2587), **activemq::commands::Response** (p. 2614), **activemq::commands::SessionInfo** (p. 2704), **activemq::commands::ShutdownInfo** (p. 2759), **activemq::commands::TransactionInfo** (p. 3040), and **activemq::commands::WireFormatInfo** (p. 3162).

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/Command.h`

6.180 activemq::state::CommandVisitor Class Reference

Interface for an Object that can visit the various Command Objects that are sent from and to this client.

```
#include <src/main/activemq/state/CommandVisitor.h>
```

Inheritance diagram for `activemq::state::CommandVisitor`:

Public Member Functions

- virtual `~CommandVisitor ()`
- virtual `decaf::lang::Pointer< commands::Command > processTransactionInfo (commands::TransactionInfo *info)=0` throw (exceptions::ActiveMQException)
- virtual `decaf::lang::Pointer< commands::Command > processRemoveInfo (commands::RemoveInfo *info)=0` throw (exceptions::ActiveMQException)
- virtual `decaf::lang::Pointer< commands::Command > processConnectionInfo (commands::ConnectionInfo *info)=0` throw (exceptions::ActiveMQException)
- virtual `decaf::lang::Pointer< commands::Command > processSessionInfo (commands::SessionInfo *info)=0` throw (exceptions::ActiveMQException)
- virtual `decaf::lang::Pointer< commands::Command > processProducerInfo (commands::ProducerInfo *info)=0` throw (exceptions::ActiveMQException)
- virtual `decaf::lang::Pointer< commands::Command > processConsumerInfo (commands::ConsumerInfo *info)=0` throw (exceptions::ActiveMQException)
- virtual `decaf::lang::Pointer< commands::Command > processRemoveConnection (commands::ConnectionId *id)=0` throw (exceptions::ActiveMQException)
- virtual `decaf::lang::Pointer< commands::Command > processRemoveSession (commands::SessionId *id)=0` throw (exceptions::ActiveMQException)
- virtual `decaf::lang::Pointer< commands::Command > processRemoveProducer (commands::ProducerId *id)=0` throw (exceptions::ActiveMQException)

- virtual **decaf::lang::Pointer< commands::Command > processRemoveConsumer** (**commands::ConsumerId *id**)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processDestinationInfo** (**commands::DestinationInfo *info**)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processRemoveDestination** (**commands::DestinationInfo *info**)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processRemoveSubscriptionInfo** (**commands::RemoveSubscriptionInfo *info**)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processMessage** (**commands::Message *send**)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processMessageAck** (**commands::MessageAck *ack**)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processMessagePull** (**commands::MessagePull *pull**)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processBeginTransaction** (**commands::TransactionInfo *info**)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processPrepareTransaction** (**commands::TransactionInfo *info**)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processCommitTransactionOnePhase** (**commands::TransactionInfo *info**)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processCommitTransactionTwoPhase** (**commands::TransactionInfo *info**)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processRollbackTransaction** (**commands::TransactionInfo *info**)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processWireFormat** (**commands::WireFormatInfo *info**)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processKeepAliveInfo** (**commands::KeepAliveInfo *info**)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processShutdownInfo** (**commands::ShutdownInfo *info**)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processFlushCommand** (**commands::FlushCommand *command**)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processBrokerInfo** (**commands::BrokerInfo *info**)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processRecoverTransactions** (**commands::TransactionInfo *info**)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processForgetTransaction** (**commands::TransactionInfo *info**)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processEndTransaction** (**commands::TransactionInfo *info**)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processMessageDispatchNotification** (**commands::MessageDispatchNotification *notification**)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processProducerAck** (**commands::ProducerAck *ack**)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processMessageDispatch** (**commands::MessageDispatch *dispatch**)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processControlCommand** (**commands::ControlCommand *command**)=0 throw (exceptions::ActiveMQException)

- virtual `decaf::lang::Pointer< commands::Command > processConnectionError (commands::ConnectionError *error)=0` throw (exceptions::ActiveMQException)
- virtual `decaf::lang::Pointer< commands::Command > processConnectionControl (commands::ConnectionControl *control)=0` throw (exceptions::ActiveMQException)
- virtual `decaf::lang::Pointer< commands::Command > processConsumerControl (commands::ConsumerControl *control)=0` throw (exceptions::ActiveMQException)
- virtual `decaf::lang::Pointer< commands::Command > processBrokerError (commands::BrokerError *error)=0` throw (exceptions::ActiveMQException)
- virtual `decaf::lang::Pointer< commands::Command > processReplayCommand (commands::ReplayCommand *replay)=0` throw (exceptions::ActiveMQException)
- virtual `decaf::lang::Pointer< commands::Command > processResponse (commands::Response *response)=0` throw (exceptions::ActiveMQException)

6.180.1 Detailed Description

Interface for an Object that can visit the various Command Objects that are sent from and to this client. The Commands themselves implement a `visit` method that is called with an instance of this interface and each one then call the appropriate `processXXX` method.

Since

3.0

6.180.2 Constructor & Destructor Documentation

- 6.180.2.1 `virtual activemq::state::CommandVisitor::~~CommandVisitor ()`
[inline, virtual]

6.180.3 Member Function Documentation

- 6.180.3.1 `virtual decaf::lang::Pointer< commands::Command > activemq::state::CommandVisitor::processBeginTransaction (commands::TransactionInfo * info)` throw (exceptions::ActiveMQException) [pure virtual]

Implemented in `activemq::state::ConnectionStateTracker` (p. 1162).

- 6.180.3.2** `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processBrokerError
(commands::BrokerError * error) throw (exceptions::ActiveMQException) [pure virtual]`
- 6.180.3.3** `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processBrokerInfo (commands::BrokerInfo * info) throw (exceptions::ActiveMQException) [pure virtual]`
- 6.180.3.4** `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processCommitTransactionOnePhase
(commands::TransactionInfo * info) throw (exceptions::ActiveMQException) [pure virtual]`

Implemented in `activemq::state::ConnectionStateTracker` (p. 1162).

- 6.180.3.5** `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processCommitTransactionTwoPhase
(commands::TransactionInfo * info) throw (exceptions::ActiveMQException) [pure virtual]`

Implemented in `activemq::state::ConnectionStateTracker` (p. 1162).

- 6.180.3.6** `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processConnectionControl
(commands::ConnectionControl * control) throw (exceptions::ActiveMQException) [pure virtual]`
- 6.180.3.7** `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processConnectionError
(commands::ConnectionError * error) throw (exceptions::ActiveMQException) [pure virtual]`
- 6.180.3.8** `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processConnectionInfo
(commands::ConnectionInfo * info) throw (exceptions::ActiveMQException) [pure virtual]`

Implemented in `activemq::state::ConnectionStateTracker` (p. 1163).

- 6.180.3.9** `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processConsumerControl
(commands::ConsumerControl * control) throw (exceptions::ActiveMQException) [pure virtual]`
- 6.180.3.10** `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processConsumerInfo
(commands::ConsumerInfo * info) throw (exceptions::ActiveMQException) [pure virtual]`

Implemented in `activemq::state::ConnectionStateTracker` (p. 1163).

6.180.3.11 virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processControlCommand
(commands::ControlCommand * *command*) throw (exceptions::ActiveMQException) [pure virtual]

6.180.3.12 virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processDestinationInfo
(commands::DestinationInfo * *info*) throw (exceptions::ActiveMQException) [pure virtual]

Implemented in `activemq::state::ConnectionStateTracker` (p. 1163).

6.180.3.13 virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processEndTransaction
(commands::TransactionInfo * *info*) throw (exceptions::ActiveMQException) [pure virtual]

Implemented in `activemq::state::ConnectionStateTracker` (p. 1163).

6.180.3.14 virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processFlushCommand
(commands::FlushCommand * *command*) throw (exceptions::ActiveMQException) [pure virtual]

6.180.3.15 virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processForgetTransaction
(commands::TransactionInfo * *info*) throw (exceptions::ActiveMQException) [pure virtual]

6.180.3.16 virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processKeepAliveInfo
(commands::KeepAliveInfo * *info*) throw (exceptions::ActiveMQException) [pure virtual]

6.180.3.17 virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processMessage (commands::Message * *send*) throw (exceptions::ActiveMQException) [pure virtual]

Implemented in `activemq::state::ConnectionStateTracker` (p. 1163).

6.180.3.18 virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processMessageAck
(commands::MessageAck * *ack*) throw (exceptions::ActiveMQException) [pure virtual]

Implemented in `activemq::state::ConnectionStateTracker` (p. 1163).

- 6.180.3.19** `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processMessageDispatch
(commands::MessageDispatch * dispatch) throw (exceptions::ActiveMQException)` [pure virtual]
- 6.180.3.20** `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processMessageDispatchNotification
(commands::MessageDispatchNotification * notification) throw (exceptions::ActiveMQException)` [pure virtual]
- 6.180.3.21** `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processMessagePull
(commands::MessagePull * pull) throw (exceptions::ActiveMQException)` [pure virtual]
- 6.180.3.22** `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processPrepareTransaction
(commands::TransactionInfo * info) throw (exceptions::ActiveMQException)` [pure virtual]

Implemented in `activemq::state::ConnectionStateTracker` (p. 1163).

- 6.180.3.23** `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processProducerAck
(commands::ProducerAck * ack) throw (exceptions::ActiveMQException)` [pure virtual]
- 6.180.3.24** `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processProducerInfo
(commands::ProducerInfo * info) throw (exceptions::ActiveMQException)` [pure virtual]

Implemented in `activemq::state::ConnectionStateTracker` (p. 1164).

- 6.180.3.25** `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processRecoverTransactions
(commands::TransactionInfo * info) throw (exceptions::ActiveMQException)` [pure virtual]
- 6.180.3.26** `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processRemoveConnection (commands::ConnectionId * id) throw (exceptions::ActiveMQException)` [pure virtual]

Implemented in `activemq::state::ConnectionStateTracker` (p. 1164).

- 6.180.3.27** `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processRemoveConsumer (commands::ConsumerId * id) throw (exceptions::ActiveMQException)` [pure virtual]

Implemented in `activemq::state::ConnectionStateTracker` (p. 1164).

6.180.3.28 virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processRemoveDestination
(commands::DestinationInfo * *info*) throw (exceptions::ActiveMQException) [pure virtual]

Implemented in `activemq::state::ConnectionStateTracker` (p. 1164).

6.180.3.29 virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processRemoveInfo
(commands::RemoveInfo * *info*) throw (exceptions::ActiveMQException) [pure virtual]

Implemented in `activemq::state::CommandVisitorAdapter` (p. 1007).

6.180.3.30 virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processRemoveProducer (commands::ProducerId * *id*) throw (exceptions::ActiveMQException) [pure virtual]

Implemented in `activemq::state::ConnectionStateTracker` (p. 1164).

6.180.3.31 virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processRemoveSession (commands::SessionId * *id*) throw (exceptions::ActiveMQException) [pure virtual]

Implemented in `activemq::state::ConnectionStateTracker` (p. 1164).

6.180.3.32 virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processRemoveSubscriptionInfo
(commands::RemoveSubscriptionInfo * *info*) throw (exceptions::ActiveMQException) [pure virtual]

6.180.3.33 virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processReplayCommand
(commands::ReplayCommand * *replay*) throw (exceptions::ActiveMQException) [pure virtual]

6.180.3.34 virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processResponse
(commands::Response * *response*) throw (exceptions::ActiveMQException) [pure virtual]

6.180.3.35 virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processRollbackTransaction
(commands::TransactionInfo * *info*) throw (exceptions::ActiveMQException) [pure virtual]

Implemented in `activemq::state::ConnectionStateTracker` (p. 1164).

6.180.3.36 `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processSessionInfo
(commands::SessionInfo * info) throw (
exceptions::ActiveMQException) [pure virtual]`

Implemented in `activemq::state::ConnectionStateTracker` (p. 1165).

6.180.3.37 `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processShutdownInfo
(commands::ShutdownInfo * info) throw (
exceptions::ActiveMQException) [pure virtual]`

6.180.3.38 `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processTransactionInfo
(commands::TransactionInfo * info) throw (
exceptions::ActiveMQException) [pure virtual]`

Implemented in `activemq::state::CommandVisitorAdapter` (p. 1008).

6.180.3.39 `virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitor::processWireFormat
(commands::WireFormatInfo * info) throw (
exceptions::ActiveMQException) [pure virtual]`

The documentation for this class was generated from the following file:

- `src/main/activemq/state/CommandVisitor.h`

6.181 `activemq::state::CommandVisitorAdapter` Class Reference

Default Implementation of a `CommandVisitor` (p. 996) that returns NULL for all calls.

`#include <src/main/activemq/state/CommandVisitorAdapter.h>`

Inheritance diagram for `activemq::state::CommandVisitorAdapter`:

Public Member Functions

- `virtual ~CommandVisitorAdapter ()`
- `virtual decaf::lang::Pointer< commands::Command > processRemoveConnection (commands::ConnectionId *id AMQCPP_UNUSED) throw (exceptions::ActiveMQException)`
- `virtual decaf::lang::Pointer< commands::Command > processRemoveSession (commands::SessionId *id AMQCPP_UNUSED) throw (exceptions::ActiveMQException)`
- `virtual decaf::lang::Pointer< commands::Command > processRemoveProducer (commands::ProducerId *id AMQCPP_UNUSED) throw (exceptions::ActiveMQException)`

- virtual **decaf::lang::Pointer< commands::Command > processRemoveConsumer** (commands::ConsumerId *id AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processDestinationInfo** (commands::DestinationInfo *info AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processRemoveDestination** (commands::DestinationInfo *info AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processRemoveSubscriptionInfo** (commands::RemoveSubscriptionInfo *info AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processMessage** (commands::Message *send AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processMessageAck** (commands::MessageAck *ack AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processMessagePull** (commands::MessagePull *pull AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processBeginTransaction** (commands::TransactionInfo *info AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processPrepareTransaction** (commands::TransactionInfo *info AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processCommitTransactionOnePhase** (commands::TransactionInfo *info AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processCommitTransactionTwoPhase** (commands::TransactionInfo *info AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processRollbackTransaction** (commands::TransactionInfo *info AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processWireFormat** (commands::WireFormatInfo *info AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processKeepAliveInfo** (commands::KeepAliveInfo *info AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processShutdownInfo** (commands::ShutdownInfo *info AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processFlushCommand** (commands::FlushCommand *command AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processBrokerInfo** (commands::BrokerInfo *info AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processRecoverTransactions** (commands::TransactionInfo *info AMQCPP_UNUSED) throw (exceptions::ActiveMQException)

- virtual **decaf::lang::Pointer**< **commands::Command** > **processForgetTransaction** (**commands::TransactionInfo** *info AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processEndTransaction** (**commands::TransactionInfo** *info AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processMessageDispatchNotification** (**commands::MessageDispatchNotification** *notification AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processProducerAck** (**commands::ProducerAck** *ack AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processMessageDispatch** (**commands::MessageDispatch** *dispatch AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processControlCommand** (**commands::ControlCommand** *command AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processConnectionError** (**commands::ConnectionError** *error AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processConnectionControl** (**commands::ConnectionControl** *control AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processConsumerControl** (**commands::ConsumerControl** *control AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processBrokerError** (**commands::BrokerError** *error AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processReplayCommand** (**commands::ReplayCommand** *replay AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processResponse** (**commands::Response** *response AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processConnectionInfo** (**commands::ConnectionInfo** *info AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processSessionInfo** (**commands::SessionInfo** *info AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processProducerInfo** (**commands::ProducerInfo** *info AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processConsumerInfo** (**commands::ConsumerInfo** *info AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processTransactionInfo** (**commands::TransactionInfo** *info) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer**< **commands::Command** > **processRemoveInfo** (**commands::RemoveInfo** *info) throw (exceptions::ActiveMQException)

6.181.1 Detailed Description

Default Implementation of a **CommandVisitor** (p. 996) that returns NULL for all calls.

Since

3.0

6.181.2 Constructor & Destructor Documentation

6.181.2.1 virtual
activemq::state::CommandVisitorAdapter::~~CommandVisitorAdapter () [inline, virtual]

6.181.3 Member Function Documentation

6.181.3.1 virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitorAdapter::processBeginTransaction (commands::TransactionInfo *info *AMQCPP_UNUSED*) throw (exceptions::ActiveMQException) [inline, virtual]

6.181.3.2 virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitorAdapter::processBrokerError (commands::BrokerError *error *AMQCPP_UNUSED*) throw (exceptions::ActiveMQException) [inline, virtual]

6.181.3.3 virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitorAdapter::processBrokerInfo (commands::BrokerInfo *info *AMQCPP_UNUSED*) throw (exceptions::ActiveMQException) [inline, virtual]

6.181.3.4 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitorAdapter::processCommitTransactionOnePhase (commands::TransactionInfo *info *AMQCPP_UNUSED*) throw (exceptions::ActiveMQException) [inline, virtual]

6.181.3.5 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitorAdapter::processCommitTransactionTwoPhase (commands::TransactionInfo *info *AMQCPP_UNUSED*) throw (exceptions::ActiveMQException) [inline, virtual]

6.181.3.6 virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitorAdapter::processConnectionControl (commands::ConnectionControl *control *AMQCPP_UNUSED*) throw (exceptions::ActiveMQException) [inline, virtual]

6.181.3.7 virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitorAdapter::processConnectionError (commands::ConnectionError *error *AMQCPP_UNUSED*) throw (exceptions::ActiveMQException) [inline, virtual]

6.181.3.8 virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitorAdapter::processConnectionInfo (commands::ConnectionInfo *info *AMQCPP_UNUSED*) throw (exceptions::ActiveMQException) [inline, virtual]

6.181.3.9 virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitorAdapter::processConsumerControl (commands::ConsumerControl *control *AMQCPP_UNUSED*) throw (exceptions::ActiveMQException) [inline, virtual]

6.181.3.10 virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitorAdapter::processConsumerInfo (commands::ConsumerInfo *info *AMQCPP_UNUSED*) throw (exceptions::ActiveMQException) [inline, virtual]

6.181.3.11 virtual decaf::lang::Pointer<commands::Command>
activemq::state::CommandVisitorAdapter::processControlCommand (commands::ControlCommand *command *AMQCPP_UNUSED*) throw (exceptions::ActiveMQException) [inline, virtual]

References `activemq::commands::ConnectionId::ID_CONNECTIONID`, `activemq::commands::ConsumerId::ID_CONSUMERID`, `activemq::commands::ProducerId::ID_PRODUCERID`, and `activemq::commands::SessionId::ID_SESSIONID`.

- 6.181.3.30 `virtual decaf::lang::Pointer<commands::Command>`
`activemq::state::CommandVisitorAdapter::processRemoveProducer`
`(commands::ProducerId *id AMQCPP_UNUSED) throw (`
`exceptions::ActiveMQException) [inline, virtual]`
- 6.181.3.31 `virtual decaf::lang::Pointer<commands::Command>`
`activemq::state::CommandVisitorAdapter::processRemoveSession`
`(commands::SessionId *id AMQCPP_UNUSED) throw (`
`exceptions::ActiveMQException) [inline, virtual]`
- 6.181.3.32 `virtual decaf::lang::Pointer<commands::Command>` `ac-`
`tivemq::state::CommandVisitorAdapter::processRemoveSubscriptionInfo`
`(commands::RemoveSubscriptionInfo *info AMQCPP_UNUSED)`
`throw (exceptions::ActiveMQException) [inline, virtual]`
- 6.181.3.33 `virtual decaf::lang::Pointer<commands::Command>`
`activemq::state::CommandVisitorAdapter::processReplayCommand`
`(commands::ReplayCommand *replay AMQCPP_UNUSED) throw (`
`exceptions::ActiveMQException) [inline, virtual]`
- 6.181.3.34 `virtual decaf::lang::Pointer<commands::Command>`
`activemq::state::CommandVisitorAdapter::processResponse`
`(commands::Response *response AMQCPP_UNUSED) throw (`
`exceptions::ActiveMQException) [inline, virtual]`
- 6.181.3.35 `virtual decaf::lang::Pointer<commands::Command>`
`activemq::state::CommandVisitorAdapter::processRollbackTransaction`
`(commands::TransactionInfo *info AMQCPP_UNUSED) throw (`
`exceptions::ActiveMQException) [inline, virtual]`
- 6.181.3.36 `virtual decaf::lang::Pointer<commands::Command>`
`activemq::state::CommandVisitorAdapter::processSessionInfo`
`(commands::SessionInfo *info AMQCPP_UNUSED) throw (`
`exceptions::ActiveMQException) [inline, virtual]`
- 6.181.3.37 `virtual decaf::lang::Pointer<commands::Command>`
`activemq::state::CommandVisitorAdapter::processShutdownInfo`
`(commands::ShutdownInfo *info AMQCPP_UNUSED) throw (`
`exceptions::ActiveMQException) [inline, virtual]`
- 6.181.3.38 `virtual decaf::lang::Pointer<commands::Command>`
`activemq::state::CommandVisitorAdapter::processTransactionInfo`
`(commands::TransactionInfo * info) throw (`
`exceptions::ActiveMQException) [inline, virtual]`

Implements `activemq::state::CommandVisitor` (p. 1003).

References `activemq::core::ActiveMQConstants::TRANSACTION_STATE_BEGIN`,
`activemq::core::ActiveMQConstants::TRANSACTION_STATE_`

```

COMMITONEPHASE,      activemq::core::ActiveMQConstants::TRANSACTION_STATE_-
COMMITTWOPHASE,      activemq::core::ActiveMQConstants::TRANSACTION_-
STATE_END,            activemq::core::ActiveMQConstants::TRANSACTION_STATE_-
FORGET,               activemq::core::ActiveMQConstants::TRANSACTION_STATE_PREPARE,
activemq::core::ActiveMQConstants::TRANSACTION_STATE_RECOVER,      and
activemq::core::ActiveMQConstants::TRANSACTION_STATE_ROLLBACK.

```

```

6.181.3.39  virtual decaf::lang::Pointer<commands::Command>
               activemq::state::CommandVisitorAdapter::processWireFormat (
               commands::WireFormatInfo *info  AMQCPP_UNUSED ) throw (
               exceptions::ActiveMQException ) [inline, virtual]

```

The documentation for this class was generated from the following file:

- src/main/activemq/state/CommandVisitorAdapter.h

6.182 decaf::lang::Comparable< T > Class Template Reference

This interface imposes a total ordering on the objects of each class that implements it.

```
#include <src/main/decaf/lang/Comparable.h>
```

Public Member Functions

- virtual **~Comparable** ()
- virtual int **compareTo** (const T &value) const =0
Compares this object with the specified object for order.
- virtual bool **equals** (const T &value) const =0
- virtual bool **operator==** (const T &value) const =0
Compares equality between this object and the one passed.
- virtual bool **operator<** (const T &value) const =0
Compares this object to another and returns true if this object is considered to be less than the one passed.

6.182.1 Detailed Description

```
template<typename T> class decaf::lang::Comparable< T >
```

This interface imposes a total ordering on the objects of each class that implements it. This ordering is referred to as the class's natural ordering, and the class's compareTo method is referred to as its natural comparison method.

6.182.2 Constructor & Destructor Documentation

6.182.2.1 `template<typename T> virtual decaf::lang::Comparable< T
>::~Comparable () [inline, virtual]`

6.182.3 Member Function Documentation

6.182.3.1 `template<typename T> virtual int decaf::lang::Comparable< T
>::compareTo (const T & value) const [pure virtual]`

Compares this object with the specified object for order.

Returns a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

In the foregoing description, the notation `sgn(expression)` designates the mathematical signum function, which is defined to return one of -1, 0, or 1 according to whether the value of expression is negative, zero or positive. The implementor must ensure `sgn(x.compareTo(y)) == -sgn(y.compareTo(x))` for all x and y. (This implies that `x.compareTo(y)` must throw an exception iff `y.compareTo(x)` throws an exception.)

The implementor must also ensure that the relation is transitive: `(x.compareTo(y)>0 && y.compareTo(z)>0)` implies `x.compareTo(z)>0`.

Finally, the implementor must ensure that `x.compareTo(y)==0` implies that `sgn(x.compareTo(z)) == sgn(y.compareTo(z))`, for all z.

It is strongly recommended, but not strictly required that `(x.compareTo(y)==0) == (x.equals(y))`. Generally speaking, any class that implements the **Comparable** (p.1009) interface and violates this condition should clearly indicate this fact. The recommended language is "Note: this class has a natural ordering that is inconsistent with equals."

Parameters

value - the Object to be compared.

Returns

a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

Implemented in **decaf::lang::Boolean** (p.676), **decaf::lang::Byte** (p.779), **decaf::lang::Character** (p.917), **decaf::lang::Double** (p.1444), **decaf::lang::Float** (p.1547), **decaf::lang::Integer** (p.1656), **decaf::lang::Long** (p.1937), and **decaf::lang::Short** (p.2732).

6.182.3.2 `template<typename T> virtual bool decaf::lang::Comparable< T
>::equals (const T & value) const [pure virtual]`

Returns

true if this value is considered equal to the passed value.

Implemented in **decaf::lang::Boolean** (p.676), **decaf::lang::Byte** (p.780), **decaf::lang::Character** (p.918), **decaf::lang::Double** (p.1446), **decaf::lang::Float** (p.1548), **decaf::lang::Integer** (p.1658), **decaf::lang::Long** (p.1939), and **decaf::lang::Short** (p.2733).

6.182.3.3 `template<typename T> virtual bool decaf::lang::Comparable< T >::operator< (const T & value) const` [pure virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

Parameters

value - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

Implemented in `decaf::lang::Boolean` (p. 676), `decaf::lang::Byte` (p. 781), `decaf::lang::Character` (p. 920), `decaf::lang::Double` (p. 1448), `decaf::lang::Float` (p. 1551), `decaf::lang::Integer` (p. 1660), `decaf::lang::Long` (p. 1941), and `decaf::lang::Short` (p. 2734).

6.182.3.4 `template<typename T> virtual bool decaf::lang::Comparable< T >::operator== (const T & value) const` [pure virtual]

Compares equality between this object and the one passed.

Parameters

value - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

Implemented in `decaf::lang::Boolean` (p. 677), `decaf::lang::Byte` (p. 781), `decaf::lang::Character` (p. 920), `decaf::lang::Double` (p. 1449), `decaf::lang::Float` (p. 1551), `decaf::lang::Integer` (p. 1660), `decaf::lang::Long` (p. 1942), and `decaf::lang::Short` (p. 2735).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Comparable.h`

6.183 `decaf::util::Comparator< T >` Class Template Reference

A comparison function, which imposes a total ordering on some collection of objects.

```
#include <src/main/decaf/util/Comparator.h>
```

Public Member Functions

- `virtual ~Comparator()`

- virtual bool **operator()** (const T &left, const T &right) const =0

*Implementation of the Binary function interface as a means of allowing a **Comparator** (p. 1011) to be passed to an STL **Map** (p. 1970) for use as the sorting criteria.*

- virtual int **compare** (const T &o1, const T &o2) const =0

Compares its two arguments for order.

6.183.1 Detailed Description

template<typename T> class decaf::util::Comparator< T >

A comparison function, which imposes a total ordering on some collection of objects. Comparators can be passed to a sort method (such as Collections.sort) to allow precise control over the sort order. Comparators can also be used to control the order of certain data structures.

The ordering imposed by a **Comparator** (p. 1011) c on a set of elements S is said to be consistent with equals if and only if (compare(e1, e2) == 0) has the same boolean value as (e1 == e2) for every e1 and e2 in S.

6.183.2 Constructor & Destructor Documentation

6.183.2.1 template<typename T> virtual decaf::util::Comparator< T >::~~Comparator () [inline, virtual]

6.183.3 Member Function Documentation

6.183.3.1 template<typename T> virtual int decaf::util::Comparator< T >::compare (const T & o1, const T & o2) const [pure virtual]

Compares its two arguments for order.

Returns a negative integer, zero, or a positive integer as the first argument is less than, equal to, or greater than the second.

The implementor must ensure that `sgn(compare(x, y)) == -sgn(compare(y, x))` for all x and y. (This implies that `compare(x, y)` must throw an exception if and only if `compare(y, x)` throws an exception.)

The implementor must also ensure that the relation is transitive: `((compare(x, y)>0) && (compare(y, z)>0))` implies `compare(x, z)>0`.

Finally, the implementer must ensure that `compare(x, y)==0` implies that `sgn(compare(x, z))==sgn(compare(y, z))` for all z.

It is generally the case, but not strictly required that `(compare(x, y)==0) == (x == y)`. Generally speaking, any comparator that violates this condition should clearly indicate this fact. The recommended language is "Note: this comparator imposes orderings that are inconsistent with equals."

Parameters

o1 - the first object to be compared

o2 - the second object to be compared

Returns

a negative integer, zero, or a positive integer as the first argument is less than, equal to, or greater than the second.

Implemented in **decaf::util::comparators::Less< E >** (p. 1863).

6.183.3.2 `template<typename T> virtual bool decaf::util::Comparator< T
>::operator() (const T & left, const T & right) const` [pure virtual]

Implementation of the Binary function interface as a means of allowing a **Comparator** (p. 1011) to be passed to an STL **Map** (p. 1970) for use as the sorting criteria.

Parameters

left - the Left hand side operand.

right - the Right hand side operand.

Returns

true if the vale of left is less than the value of right.

Implemented in **decaf::util::comparators::Less< E >** (p. 1864).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/Comparator.h`

6.184 activemq::util::CompositeData Class Reference

Represents a Composite URI.

```
#include <src/main/activemq/util/CompositeData.h>
```

Public Member Functions

- **CompositeData** ()
- virtual **~CompositeData** ()
- **StlList< URI > &getComponents** ()
- const **StlList< URI > &getComponents** () const
- void **setComponents** (const **StlList< URI > &components**)
- std::string **getFragment** () const
- void **setFragment** (const std::string &fragment)
- const **Properties &getParameters** () const
- void **setParameters** (const **Properties ¶meters**)
- std::string **getScheme** () const
- void **setScheme** (const std::string &scheme)
- std::string **getPath** () const
- void **setPath** (const std::string &path)
- std::string **getHost** () const
- void **setHost** (const std::string &host)
- **URI toURI** () const throw (decaf::net::URISyntaxException)

6.184.1 Detailed Description

Represents a Composite URI.

Since

3.0

6.184.2 Constructor & Destructor Documentation

6.184.2.1 `activemq::util::CompositeData::CompositeData ()`

6.184.2.2 `virtual activemq::util::CompositeData::~~CompositeData () [virtual]`

6.184.3 Member Function Documentation

6.184.3.1 `StlList<URI>& activemq::util::CompositeData::getComponents () [inline]`

6.184.3.2 `const StlList<URI>& activemq::util::CompositeData::getComponents () const [inline]`

6.184.3.3 `std::string activemq::util::CompositeData::getFragment () const [inline]`

6.184.3.4 `std::string activemq::util::CompositeData::getHost () const [inline]`

6.184.3.5 `const Properties& activemq::util::CompositeData::getParameters () const [inline]`

6.184.3.6 `std::string activemq::util::CompositeData::getPath () const [inline]`

6.184.3.7 `std::string activemq::util::CompositeData::getScheme () const [inline]`

6.184.3.8 `void activemq::util::CompositeData::setComponents (const StlList<URI> & components) [inline]`

6.184.3.9 `void activemq::util::CompositeData::setFragment (const std::string & fragment) [inline]`

6.184.3.10 `void activemq::util::CompositeData::setHost (const std::string & host) [inline]`

6.184.3.11 `void activemq::util::CompositeData::setParameters (const Properties & parameters) [inline]`

6.184.3.12 `void activemq::util::CompositeData::setPath (const std::string & path) [inline]`

6.184.3.13 `void activemq::util::CompositeData::setScheme (const std::string & scheme) [inline]`

6.184.3.14 `URI activemq::util::CompositeData::toURI () const throw (decaf::net::URISyntaxException)`

The documentation for this class was generated from the following file:

- `src/main/activemq/util/CompositeData.h`

6.185 activemq::threads::CompositeTask Class Reference

Represents a single task that can be part of a set of Tasks that are contained in a CompositeTaskRunner (p.1017).

```
#include <src/main/activemq/threads/CompositeTask.h>
```

Inheritance diagram for activemq::threads::CompositeTask:

Public Member Functions

- virtual `~CompositeTask()`
- virtual bool `isPending()` const =0

*Indicates whether this task has any pending work that needs to be done, if not then it is skipped and the next **Task** (p.2956) in the CompositeTaskRunner's list of tasks is checked, if none of the tasks have any pending work to do, then the runner can go to sleep until it awakened by a call to `wakeup`.*

6.185.1 Detailed Description

Represents a single task that can be part of a set of Tasks that are contained in a CompositeTaskRunner (p.1017).

Since

3.0

6.185.2 Constructor & Destructor Documentation

6.185.2.1 virtual `activemq::threads::CompositeTask::~~CompositeTask()`
[inline, virtual]

6.185.3 Member Function Documentation

6.185.3.1 virtual bool `activemq::threads::CompositeTask::isPending()` const
[pure virtual]

Indicates whether this task has any pending work that needs to be done, if not then it is skipped and the next **Task** (p.2956) in the CompositeTaskRunner's list of tasks is checked, if none of the tasks have any pending work to do, then the runner can go to sleep until it awakened by a call to `wakeup`.

Since

3.0

Implemented in `activemq::transport::failover::BackupTransportPool` (p.596), `activemq::transport::failover::CloseTransportsTask` (p.953), and `activemq::transport::failover::FailoverTransport` (p.1518).

The documentation for this class was generated from the following file:

- `src/main/activemq/threads/CompositeTask.h`

6.186 `activemq::threads::CompositeTaskRunner` Class Reference

A **Task** (p. 2956) Runner that can contain one or more `CompositeTasks` that are each checked for pending work and run if any is present in the order that the tasks were added.

```
#include <src/main/activemq/threads/CompositeTaskRunner.h>
```

Inheritance diagram for `activemq::threads::CompositeTaskRunner`:

Public Member Functions

- **CompositeTaskRunner** ()
- virtual **~CompositeTaskRunner** ()
- void **addTask** (`CompositeTask *task`)
*Adds a new **CompositeTask** (p. 1016) to the Set of Tasks that this class manages.*
- void **removeTask** (`CompositeTask *task`)
*Removes a **CompositeTask** (p. 1016) that was added previously.*
- virtual void **shutdown** (unsigned int timeout)
Shutdown after a timeout, does not guarantee that the task's iterate method has completed and the thread halted.
- virtual void **shutdown** ()
Shutdown once the task has finished and the TaskRunner's thread has exited.
- virtual void **wakeup** ()
*Signal the **TaskRunner** (p. 2958) to wakeup and execute another iteration cycle on the task, the **Task** (p. 2956) instance will be run until its iterate method has returned false indicating it is done.*

Protected Member Functions

- virtual void **run** ()
Run method - called by the Thread class in the context of the thread.
- virtual bool **iterate** ()

6.186.1 Detailed Description

A **Task** (p. 2956) Runner that can contain one or more `CompositeTasks` that are each checked for pending work and run if any is present in the order that the tasks were added.

Since

3.0

6.186.2 Constructor & Destructor Documentation

6.186.2.1 `activemq::threads::CompositeTaskRunner::CompositeTaskRunner ()`

6.186.2.2 `virtual
activemq::threads::CompositeTaskRunner::~~CompositeTaskRunner ()
[virtual]`

6.186.3 Member Function Documentation

6.186.3.1 `void activemq::threads::CompositeTaskRunner::addTask (
CompositeTask * task)`

Adds a new **CompositeTask** (p. 1016) to the Set of Tasks that this class manages.

Parameters

task - Pointer to a **CompositeTask** (p. 1016) instance.

6.186.3.2 `virtual bool activemq::threads::CompositeTaskRunner::iterate ()
[protected, virtual]`

6.186.3.3 `void activemq::threads::CompositeTaskRunner::removeTask (
CompositeTask * task)`

Removes a **CompositeTask** (p. 1016) that was added previously.

Parameters

task - Pointer to a **CompositeTask** (p. 1016) instance.

6.186.3.4 `virtual void activemq::threads::CompositeTaskRunner::run ()
[protected, virtual]`

Run method - called by the Thread class in the context of the thread.

Implements **decaf::lang::Runnable** (p. 2642).

6.186.3.5 `virtual void activemq::threads::CompositeTaskRunner::shutdown ()
[virtual]`

Shutdown once the task has finished and the TaskRunner's thread has exited.

6.186.3.6 `virtual void activemq::threads::CompositeTaskRunner::shutdown (
unsigned int timeout) [virtual]`

Shutdown after a timeout, does not guarantee that the task's iterate method has completed and the thread halted.

Parameters

timeout - Time in Milliseconds to wait for the task to stop.

6.186.3.7 virtual void activemq::threads::CompositeTaskRunner::wakeup () [virtual]

Signal the **TaskRunner** (p. 2958) to wakeup and execute another iteration cycle on the task, the **Task** (p. 2956) instance will be run until its iterate method has returned false indicating it is done.

The documentation for this class was generated from the following file:

- src/main/activemq/threads/**CompositeTaskRunner.h**

6.187 activemq::transport::CompositeTransport Class Reference

A Composite **Transport** (p. 3066) is a **Transport** (p. 3066) implementation that is composed of several Transports.

```
#include <src/main/activemq/transport/CompositeTransport.h>
```

Inheritance diagram for activemq::transport::CompositeTransport:

Public Member Functions

- virtual ~**CompositeTransport** ()
- virtual void **addURI** (const **List**< **URI** > &uris)=0

*Add a URI to the list of URI's that will represent the set of Transports that this **Transport** (p. 3066) is a composite of.*

- virtual void **removeURI** (const **List**< **URI** > &uris)=0

*Remove a URI from the set of URI's that represents the set of Transports that this **Transport** (p. 3066) is composed of, removing a URI for which the composite has created a connected **Transport** (p. 3066) should result in that **Transport** (p. 3066) being disposed of.*

6.187.1 Detailed Description

A Composite **Transport** (p. 3066) is a **Transport** (p. 3066) implementation that is composed of several Transports. The composition could be such that only one **Transport** (p. 3066) exists for each URI that is composed or there could be many active Transports working at once.

Since

3.0

6.187.2 Constructor & Destructor Documentation

6.187.2.1 virtual activemq::transport::CompositeTransport::~CompositeTransport () [inline, virtual]

6.187.3 Member Function Documentation

6.187.3.1 virtual void activemq::transport::CompositeTransport::addURI (const List< URI > & *uris*) [pure virtual]

Add a URI to the list of URI's that will represent the set of Transports that this **Transport** (p. 3066) is a composite of.

Parameters

uris The new URI set to add to the set this composite maintains.

6.187.3.2 virtual void activemq::transport::CompositeTransport::removeURI (const List< URI > & *uris*) [pure virtual]

Remove a URI from the set of URI's that represents the set of Transports that this **Transport** (p. 3066) is composed of, removing a URI for which the composite has created a connected **Transport** (p. 3066) should result in that **Transport** (p. 3066) being disposed of.

Parameters

uris The new URI set to remove to the set this composite maintains.

The documentation for this class was generated from the following file:

- src/main/activemq/transport/**CompositeTransport.h**

6.188 decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR > Class Template Reference

Interface for a **Map** (p. 1970) type that provides additional atomic putIfAbsent, remove, and replace methods alongside the already available **Map** (p. 1970) interface.

#include <src/main/decaf/util/concurrent/ConcurrentMap.h>

Inheritance diagram for decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR >:

Public Member Functions

- virtual ~**ConcurrentMap** ()
- virtual bool **putIfAbsent** (const K &key, const V &value)=0 throw (decaf::lang::exceptions::UnsupportedOperationException)

If the specified key is not already associated with a value, associate it with the given value.

- virtual bool **remove** (const K &key, const V &value)=0
Remove entry for key only if currently mapped to given value.
- virtual bool **replace** (const K &key, const V &oldValue, const V &newValue)=0
Replace entry for key only if currently mapped to given value.
- virtual V **replace** (const K &key, const V &value)=0 throw (decaf::lang::exceptions::NoSuchElementException)
Replace entry for key only if currently mapped to some value.

6.188.1 Detailed Description

template<typename K, typename V, typename COMPARATOR> class decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR >

Interface for a **Map** (p.1970) type that provides additional atomic putIfAbsent, remove, and replace methods alongside the already available **Map** (p.1970) interface.

Since

1.0

6.188.2 Constructor & Destructor Documentation

6.188.2.1 template<typename K, typename V, typename COMPARATOR>
virtual decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR
>::~ConcurrentMap () [inline, virtual]

6.188.3 Member Function Documentation

6.188.3.1 template<typename K, typename V, typename COMPARATOR>
virtual bool decaf::util::concurrent::ConcurrentMap< K, V,
COMPARATOR >::putIfAbsent (const K & *key*, const V & *value*)
throw (decaf::lang::exceptions::UnsupportedOperationException) [pure
virtual]

If the specified key is not already associated with a value, associate it with the given value.

This is equivalent to

```
if( !map.containsKey( key ) ) {
    map.put( key, value );
    return true;
} else {
    return false;
}
```

except that the action is performed atomically.

Parameters

key The key to map the value to.

value The value to map to the given key.

Returns

true if the put operation was performed otherwise return false which indicates there was a value previously mapped to the key.

Exceptions

UnsupportedOperationException if the put operation is not supported by this map

Implemented in decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR > (p.1035), decaf::util::concurrent::ConcurrentStlMap< Pointer< TransactionId >, Pointer< TransactionState >, TransactionId::COMPARATOR > (p.1035), decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR > (p.1035), decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR > (p.1035), decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR > (p.1035), decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR > (p.1035), and decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR > (p.1035).

6.188.3.2 template<typename K, typename V, typename COMPARATOR> virtual
bool decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR
>::remove (const K & *key*, const V & *value*) [pure virtual]

Remove entry for key only if currently mapped to given value.

Acts as

```
if( ( map.containsKey( key ) && ( map.get( key ) == value ) ) ) {
    map.remove( key );
    return true;
} else {
    return false;
}
```

except that the action is performed atomically.

Parameters

key key with which the specified value is associated.

value value associated with the specified key.

Returns

true if the value was removed, false otherwise

Implemented in decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR > (p.1035), decaf::util::concurrent::ConcurrentStlMap<

`Pointer< TransactionId >, Pointer< TransactionState >, TransactionId::COMPARATOR >` (p. 1035), `decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >` (p. 1035), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >` (p. 1035), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >` (p. 1035), `decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >` (p. 1035), and `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >` (p. 1035).

6.188.3.3 `template<typename K, typename V, typename COMPARATOR> virtual bool decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR >::replace (const K & key, const V & oldValue, const V & newValue)` [pure virtual]

Replace entry for key only if currently mapped to given value.

Acts as

```
if( ( map.containsKey( key ) && ( map.get( key ) == oldValue ) ) ) {
    map.put( key, newValue );
    return true;
} else {
    return false;
}
```

except that the action is performed atomically.

Parameters

key key with which the specified value is associated.

oldValue value expected to be associated with the specified key.

newValue value to be associated with the specified key.

Returns

true if the value was replaced

Implemented in `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >` (p. 1037), `decaf::util::concurrent::ConcurrentStlMap< Pointer< TransactionId >, Pointer< TransactionState >, TransactionId::COMPARATOR >` (p. 1037), `decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >` (p. 1037), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >` (p. 1037), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >` (p. 1037), `decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >` (p. 1037), and `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >` (p. 1037).

6.188.3.4 `template<typename K, typename V, typename COMPARATOR> virtual
 V decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR
 >::replace (const K & key, const V & value) throw (
 decaf::lang::exceptions::NoSuchElementException) [pure virtual]`

Replace entry for key only if currently mapped to some value.

Acts as

```
if( ( map.containsKey( key ) ) ) {
    return map.put( key, value );
} else {
    throw NoSuchElementException(...);
};
```

except that the action is performed atomically.

Parameters

key key with which the specified value is associated.

value value to be associated with the specified key.

Returns

copy of the previous value associated with specified key, or throws an NoSuchElementException if there was no mapping for key.

Exceptions

NoSuchElementException if there was no previous mapping.

Implemented in `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >` (p. 1036), `decaf::util::concurrent::ConcurrentStlMap< Pointer< TransactionId >, Pointer< TransactionState >, TransactionId::COMPARATOR >` (p. 1036), `decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >` (p. 1036), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >` (p. 1036), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >` (p. 1036), `decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >` (p. 1036), and `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >` (p. 1036).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/ConcurrentMap.h`

6.189 decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR > Class Template Reference

Map (p. 1970) template that wraps around a `std::map` to provide a more user-friendly interface and to provide common functions that do not exist in `std::map`.

```
#include <src/main/decaf/util/concurrent/ConcurrentStlMap.h>
```

Inheritance diagram for decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >:

Public Member Functions

- **ConcurrentStlMap** ()
Default constructor - does nothing.
- **ConcurrentStlMap** (const **ConcurrentStlMap** &source)
Copy constructor - copies the content of the given map into this one.
- **ConcurrentStlMap** (const **Map**< K, V, COMPARATOR > &source)
Copy constructor - copies the content of the given map into this one.
- virtual ~**ConcurrentStlMap** ()
- virtual bool **equals** (const **ConcurrentStlMap** &source) const
- virtual bool **equals** (const **Map**< K, V, COMPARATOR > &source) const
Comparison, equality is dependent on the method of determining if the element are equal.
- virtual void **copy** (const **ConcurrentStlMap** &source)
- virtual void **copy** (const **Map**< K, V, COMPARATOR > &source)
Copies the content of the source map into this map.
- virtual void **clear** () throw (decaf::lang::exceptions::UnsupportedOperationException)
Removes all keys and values from this map.
Exceptions
***UnsupportedOperationException** if this map is unmodifiable.*
- virtual bool **containsKey** (const K &key) const
Indicates whether or this map contains a value for the given key.
Parameters
***key** The key to look up.*
Returns
true if this map contains the value, otherwise false.
- virtual bool **containsValue** (const V &value) const
Indicates whether or this map contains a value for the given value, i.e. they are equal, this is done by operator== so the types must pass equivalence testing in this manner.
Parameters
***value** The Value to look up.*
Returns
true if this map contains the value, otherwise false.

- virtual bool **isEmpty** () const
Returns
*if the **Map** (p. 1970) contains any element or not, TRUE or FALSE*

- virtual std::size_t **size** () const
Returns
The number of elements (key/value pairs) in this map.

- virtual V & **get** (const K &key) throw (lang::exceptions::NoSuchElementException)
*Gets the value mapped to the specified key in the **Map** (p. 1970).*
*If there is no element in the map whose key is equivalent to the key provided then a *NoSuchElementException* is thrown.*
Parameters
key The search key.
Returns
A reference to the value for the given key.
Exceptions
***NoSuchElementException** if the key requests doesn't exist in the **Map** (p. 1970).*

- virtual const V & **get** (const K &key) const throw (lang::exceptions::NoSuchElementException)
*Gets the value mapped to the specified key in the **Map** (p. 1970).*
*If there is no element in the map whose key is equivalent to the key provided then a *NoSuchElementException* is thrown.*
Parameters
key The search key.
Returns
A {const} reference to the value for the given key.
Exceptions
***NoSuchElementException** if the key requests doesn't exist in the **Map** (p. 1970).*

- virtual void **put** (const K &key, const V &value) throw (decaf::lang::exceptions::UnsupportedOperationException)
Sets the value for the specified key.
Parameters
key The target key.
value The value to be set.
Exceptions
***UnsupportedOperationException** if this map is unmodifiable.*

- virtual void **putAll** (const ConcurrentStlMap< K, V, COMPARATOR > &other) throw (decaf::lang::exceptions::UnsupportedOperationException)

- virtual void **putAll** (const Map< K, V, COMPARATOR > &other) throw (decaf::lang::exceptions::UnsupportedOperationException)
*Stores a copy of the Mappings contained in the other **Map** (p. 1970) in this one.*

Parameters

*other A **Map** (p. 1970) instance whose elements are to all be inserted in this **Map** (p. 1970).*

Exceptions

UnsupportedOperationException *If the implementing class does not support the putAll operation.*

- virtual V **remove** (const K &key) throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException)

Removes the value (key/value pair) for the specified key from the map, returns a copy of the value that was mapped to the key.

Parameters

key The search key.

Returns

a copy of the element that was previously mapped to the given key

Exceptions

NoSuchElementException *if this key is not in the **Map** (p. 1970).*
UnsupportedOperationException *if this map is unmodifiable.*

- virtual std::vector< K > **keySet** () const

*Returns a **Set** (p. 2729) view of the mappings contained in this map.*

*The set is backed by the map, so changes to the map are reflected in the set, and vice-versa. If the map is modified while an iteration over the set is in progress (except through the iterator's own remove operation, or through the setValue operation on a map entry returned by the iterator) the results of the iteration are undefined. The set supports element removal, which removes the corresponding mapping from the map, via the **Iterator.remove** (p. 1718), **Set.remove** (p. 127), removeAll, retainAll and clear operations. It does not support the add or addAll operations.*

Returns

the entire set of keys in this map as a std::vector.

- virtual std::vector< V > **values** () const

Returns

the entire set of values in this map as a std::vector.

- bool **putIfAbsent** (const K &key, const V &value) throw (decaf::lang::exceptions::UnsupportedOperationException)

If the specified key is not already associated with a value, associate it with the given value.

- bool **remove** (const K &key, const V &value)

Remove entry for key only if currently mapped to given value.

- bool **replace** (const K &key, const V &oldValue, const V &newValue)

Replace entry for key only if currently mapped to given value.

- V **replace** (const K &key, const V &value) throw (decaf::lang::exceptions::NoSuchElementException)

Replace entry for key only if currently mapped to some value.

- virtual void **lock** () throw (decaf::lang::exceptions::RuntimeException)

Locks the object.

- virtual bool **tryLock** () throw (decaf::lang::exceptions::RuntimeException)
*Attempts to **Lock** (p.1903) the object, if the lock is already held by another thread than this method returns false.*
- virtual void **unlock** () throw (decaf::lang::exceptions::RuntimeException)
Unlocks the object.
- virtual void **wait** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (long long millisecs) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (long long millisecs, int nanos) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **notify** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)
Signals a waiter on this object that it can now wake up and continue.
- virtual void **notifyAll** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)
Signals the waiters on this object that it can now wake up and continue.

6.189.1 Detailed Description

```
template<typename K, typename V, typename COMPARATOR = std::less<K>>
class decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >
```

Map (p.1970) template that wraps around a std::map to provide a more user-friendly interface and to provide common functions that do not exist in std::map. This version of **Map** (p.1970) extends the **ConcurrentMap** (p.1020) interface and implements all the methods defined in that interface. Unlike a Java ConcurrentHashMap this implementations synchronizes all methods such that any call to this class will block if another thread is already holding a lock, much like the Java HashTable.

Since

1.0

6.189.2 Constructor & Destructor Documentation

6.189.2.1 `template<typename K, typename V, typename COMPARATOR
= std::less<K>> decaf::util::concurrent::ConcurrentStlMap< K, V,
COMPARATOR >::ConcurrentStlMap () [inline]`

Default constructor - does nothing.

6.189.2.2 `template<typename K, typename V, typename COMPARATOR
= std::less<K>> decaf::util::concurrent::ConcurrentStlMap< K, V,
COMPARATOR >::ConcurrentStlMap (const ConcurrentStlMap< K,
V, COMPARATOR > & source) [inline]`

Copy constructor - copies the content of the given map into this one.

Parameters

source The source map.

6.189.2.3 `template<typename K, typename V, typename COMPARATOR
= std::less<K>> decaf::util::concurrent::ConcurrentStlMap< K,
V, COMPARATOR >::ConcurrentStlMap (const Map< K, V,
COMPARATOR > & source) [inline]`

Copy constructor - copies the content of the given map into this one.

Parameters

source The source map.

6.189.2.4 `template<typename K, typename V, typename COMPARATOR =
std::less<K>> virtual decaf::util::concurrent::ConcurrentStlMap< K, V,
COMPARATOR >::~~ConcurrentStlMap () [inline, virtual]`

6.189.3 Member Function Documentation

6.189.3.1 `template<typename K, typename V, typename COMPARATOR =
std::less<K>> virtual void decaf::util::concurrent::ConcurrentStlMap<
K, V, COMPARATOR >::clear () throw (
decaf::lang::exceptions::UnsupportedOperationException) [inline,
virtual]`

Removes all keys and values from this map.

Exceptions

UnsupportedOperationException if this map is unmodifiable.

Implements `decaf::util::Map< K, V, COMPARATOR >` (p.1972).

Referenced by `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::copy()`.

6.189.3.2 `template<typename K, typename V, typename COMPARATOR =
 std::less<K>> virtual bool decaf::util::concurrent::ConcurrentStlMap<
 K, V, COMPARATOR >::containsKey (const K & key) const
 [inline, virtual]`

Indicates whether or this map contains a value for the given key.

Parameters

key The key to look up.

Returns

true if this map contains the value, otherwise false.

Implements `decaf::util::Map< K, V, COMPARATOR >` (p.1973).

Referenced by `decaf::util::concurrent::ConcurrentStlMap< Pointer< Produc-
 erId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::equals()`,
`decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState
 >, ProducerId::COMPARATOR >::putIfAbsent()`, `decaf::util::concurrent::ConcurrentStlMap<
 Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::remove()`,
 and `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState
 >, ProducerId::COMPARATOR >::replace()`.

6.189.3.3 `template<typename K, typename V, typename COMPARATOR =
 std::less<K>> virtual bool decaf::util::concurrent::ConcurrentStlMap<
 K, V, COMPARATOR >::containsValue (const V & value) const
 [inline, virtual]`

Indicates whether or this map contains a value for the given value, i.e.

they are equal, this is done by operator== so the types must pass equivalence testing in this
 manner.

Parameters

value The Value to look up.

Returns

true if this map contains the value, otherwise false.

Implements `decaf::util::Map< K, V, COMPARATOR >` (p.1973).

6.189.3.4 `template<typename K, typename V, typename COMPARATOR =
 std::less<K>> virtual void decaf::util::concurrent::ConcurrentStlMap<
 K, V, COMPARATOR >::copy (const ConcurrentStlMap< K, V,
 COMPARATOR > & source) [inline, virtual]`

Referenced by `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< Pro-
 ducerState >, ProducerId::COMPARATOR >::ConcurrentStlMap()`.

6.189.3.5 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::copy (const Map< K, V, COMPARATOR > & source) [inline, virtual]`

Copies the content of the source map into this map.

Erases all existing data in this map.

Parameters

source The source object to copy from.

Implements `decaf::util::Map< K, V, COMPARATOR >` (p. 1974).

6.189.3.6 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual bool decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::equals (const Map< K, V, COMPARATOR > & source) const [inline, virtual]`

Comparison, equality is dependent on the method of determining if the element are equal.

Parameters

source - `Map` (p. 1970) to compare to this one.

Returns

true if the `Map` (p. 1970) passed is equal in value to this one.

Implements `decaf::util::Map< K, V, COMPARATOR >` (p. 1974).

6.189.3.7 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual bool decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::equals (const ConcurrentStlMap< K, V, COMPARATOR > & source) const [inline, virtual]`

6.189.3.8 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual V& decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::get (const K & key) throw (lang::exceptions::NoSuchElementException) [inline, virtual]`

Gets the value mapped to the specified key in the `Map` (p. 1970).

If there is no element in the map whose key is equivalent to the key provided then a `NoSuchElementException` is thrown.

Parameters

key The search key.

Returns

A reference to the value for the given key.

Exceptions

NoSuchElementException if the key requests doesn't exist in the **Map** (p. 1970).

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 1975).

6.189.3.9 `template<typename K, typename V, typename
COMPARATOR = std::less<K>> virtual const V&
decaf::util::concurrent::ConcurrentStlMap< K, V,
COMPARATOR >::get (const K & key) const throw (
lang::exceptions::NoSuchElementException) [inline, virtual]`

Gets the value mapped to the specified key in the **Map** (p. 1970).

If there is no element in the map whose key is equivalent to the key provided then a *NoSuchElementException* is thrown.

Parameters

key The search key.

Returns

A {const} reference to the value for the given key.

Exceptions

NoSuchElementException if the key requests doesn't exist in the **Map** (p. 1970).

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 1976).

6.189.3.10 `template<typename K, typename V, typename COMPARATOR =
std::less<K>> virtual bool decaf::util::concurrent::ConcurrentStlMap<
K, V, COMPARATOR >::isEmpty () const [inline, virtual]`

Returns

if the **Map** (p. 1970) contains any element or not, TRUE or FALSE

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 1977).

6.189.3.11 `template<typename K, typename V, typename
COMPARATOR = std::less<K>> virtual std::vector<K>
decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR
>::keySet () const [inline, virtual]`

Returns a **Set** (p. 2729) view of the mappings contained in this map.

The set is backed by the map, so changes to the map are reflected in the set, and vice-versa. If the map is modified while an iteration over the set is in progress (except through the iterator's own remove operation, or through the setValue operation on a map entry returned by the iterator) the results of the iteration are undefined. The set supports element removal, which removes the corresponding mapping from the map, via the **Iterator.remove** (p. 1718), **Set.remove** (p. 127), **removeAll**, **retainAll** and **clear** operations. It does not support the **add** or **addAll** operations.

Returns

the entire set of keys in this map as a `std::vector`.

Implements `decaf::util::Map< K, V, COMPARATOR >` (p.1977).

6.189.3.12 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::lock () throw (decaf::lang::exceptions::RuntimeException) [inline, virtual]`

Locks the object.

Exceptions

RuntimeException if an error occurs while locking the object.

Implements `decaf::util::concurrent::Synchronizable` (p.2932).

6.189.3.13 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::notify () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException) [inline, virtual]`

Signals a waiter on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

IllegalMonitorStateException - if the current thread is not the owner of the the **Synchronizable** (p.2930) Object.

RuntimeException if an error occurs while notifying one of the waiting threads.

Implements `decaf::util::concurrent::Synchronizable` (p.2933).

6.189.3.14 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::notifyAll () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException) [inline, virtual]`

Signals the waiters on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

IllegalMonitorStateException - if the current thread is not the owner of the the **Synchronizable** (p.2930) Object.

RuntimeException if an error occurs while notifying the waiting threads.

Implements `decaf::util::concurrent::Synchronizable` (p.2934).

6.189.3.15 `template<typename K, typename V, typename COMPARATOR =
 std::less<K>> virtual void decaf::util::concurrent::ConcurrentStlMap<
 K, V, COMPARATOR >::put (const K & key, const V & value
) throw (decaf::lang::exceptions::UnsupportedOperationException)
 [inline, virtual]`

Sets the value for the specified key.

Parameters

key The target key.

value The value to be set.

Exceptions

UnsupportedOperationException if this map is unmodifiable.

Implements `decaf::util::Map< K, V, COMPARATOR >` (p. 1978).

Referenced by `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::putAll()`, `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::putIfAbsent()`, and `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::replace()`.

6.189.3.16 `template<typename K, typename V, typename COMPARATOR =
 std::less<K>> virtual void decaf::util::concurrent::ConcurrentStlMap<
 K, V, COMPARATOR >::putAll (const Map<
 K, V, COMPARATOR > & other) throw (
 decaf::lang::exceptions::UnsupportedOperationException) [inline,
 virtual]`

Stores a copy of the Mappings contained in the other **Map** (p. 1970) in this one.

Parameters

other A **Map** (p. 1970) instance whose elements are to all be inserted in this **Map** (p. 1970).

Exceptions

UnsupportedOperationException If the implementing class does not support the putAll operation.

Implements `decaf::util::Map< K, V, COMPARATOR >` (p. 1979).

6.189.3.17 `template<typename K, typename V, typename COMPARATOR =
 std::less<K>> virtual void decaf::util::concurrent::ConcurrentStlMap<
 K, V, COMPARATOR >::putAll (const ConcurrentStlMap<
 K, V, COMPARATOR > & other) throw (
 decaf::lang::exceptions::UnsupportedOperationException) [inline,
 virtual]`

Referenced by `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::copy()`.

6.189.3.18 `template<typename K, typename V, typename COMPARATOR = std::less<K>> bool decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::putIfAbsent (const K & key, const V & value) throw (decaf::lang::exceptions::UnsupportedOperationException) [inline, virtual]`

If the specified key is not already associated with a value, associate it with the given value.

This is equivalent to

```
if( !map.containsKey( key ) ) {
    map.put( key, value );
    return true;
} else {
    return false;
}
```

except that the action is performed atomically.

Parameters

key The key to map the value to.

value The value to map to the given key.

Returns

true if the put operation was performed otherwise return false which indicates there was a value previously mapped to the key.

Exceptions

UnsupportedOperationException if the put operation is not supported by this map

Implements `decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR >` (p. 1021).

6.189.3.19 `template<typename K, typename V, typename COMPARATOR = std::less<K>> bool decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::remove (const K & key, const V & value) [inline, virtual]`

Remove entry for key only if currently mapped to given value.

Acts as

```
if( map.containsKey( key ) && ( map.get( key ) == value ) ) {
    map.remove( key );
    return true;
} else {
    return false;
}
```

except that the action is performed atomically.

Parameters

key key with which the specified value is associated.
value value associated with the specified key.

Returns

true if the value was removed, false otherwise

Implements **decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR >** (p. 1022).

6.189.3.20 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual V decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::remove (const K & key) throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException) [inline, virtual]`

Removes the value (key/value pair) for the specified key from the map, returns a copy of the value that was mapped to the key.

Parameters

key The search key.

Returns

a copy of the element that was previously mapped to the given key

Exceptions

NoSuchElementException if this key is not in the **Map** (p. 1970).

UnsupportedOperationException if this map is unmodifiable.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 1980).

6.189.3.21 `template<typename K, typename V, typename COMPARATOR = std::less<K>> V decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::replace (const K & key, const V & value) throw (decaf::lang::exceptions::NoSuchElementException) [inline, virtual]`

Replace entry for key only if currently mapped to some value.

Acts as

```
if( map.containsKey( key ) ) {
    return map.put( key, value );
} else {
    throw NoSuchElementException(...);
};
```

except that the action is performed atomically.

Parameters

key key with which the specified value is associated.

value value to be associated with the specified key.

Returns

copy of the previous value associated with specified key, or throws an `NoSuchElementException` if there was no mapping for key.

Exceptions

NoSuchElementException if there was no previous mapping.

Implements `decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR >` (p. 1024).

6.189.3.22 `template<typename K, typename V, typename COMPARATOR = std::less<K>> bool decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::replace (const K & key, const V & oldValue, const V & newValue) [inline, virtual]`

Replace entry for key only if currently mapped to given value.

Acts as

```
if( map.containsKey( key ) && ( map.get( key ) == oldValue ) ) {
    map.put( key, newValue );
    return true;
} else {
    return false;
}
```

except that the action is performed atomically.

Parameters

key key with which the specified value is associated.

oldValue value expected to be associated with the specified key.

newValue value to be associated with the specified key.

Returns

true if the value was replaced

Implements `decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR >` (p. 1023).

6.189.3.23 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual std::size_t decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::size () const [inline, virtual]`

Returns

The number of elements (key/value pairs) in this map.

Implements `decaf::util::Map< K, V, COMPARATOR >` (p.1981).

6.189.3.24 `template<typename K, typename V, typename COMPARATOR =
 std::less<K>> virtual bool decaf::util::concurrent::ConcurrentStlMap<
 K, V, COMPARATOR >::tryLock () throw (
 decaf::lang::exceptions::RuntimeException) [inline, virtual]`

Attempts to **Lock** (p.1903) the object, if the lock is already held by another thread than this method returns false.

Returns

true if the lock was acquired, false if it is already held by another thread.

Exceptions

RuntimeException if an error occurs while locking the object.

Implements `decaf::util::concurrent::Synchronizable` (p.2935).

6.189.3.25 `template<typename K, typename V, typename COMPARATOR =
 std::less<K>> virtual void decaf::util::concurrent::ConcurrentStlMap<
 K, V, COMPARATOR >::unlock () throw (
 decaf::lang::exceptions::RuntimeException) [inline, virtual]`

Unlocks the object.

Exceptions

RuntimeException if an error occurs while unlocking the object.

Implements `decaf::util::concurrent::Synchronizable` (p.2936).

6.189.3.26 `template<typename K, typename V, typename
 COMPARATOR = std::less<K>> virtual std::vector<V>
 decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR
 >::values () const [inline, virtual]`

Returns

the entire set of values in this map as a `std::vector`.

Implements `decaf::util::Map< K, V, COMPARATOR >` (p.1981).

Referenced by `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::values()`.

6.189.3.27 `template<typename K, typename V, typename COMPARATOR =
 std::less<K>> virtual void decaf::util::concurrent::ConcurrentStlMap<
 K, V, COMPARATOR >::wait (long long millisecs
) throw (decaf::lang::exceptions::RuntimeException,
 decaf::lang::exceptions::IllegalMonitorStateException,
 decaf::lang::exceptions::InterruptedException) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to `Notify`.

Must have this object locked before calling. This wait will timeout after the specified time interval.

Parameters

millisecs the time in milliseconds to wait, or WAIT_INFINITE

Exceptions

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the **Synchronizable** (p. 2930) Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2938).

```
6.189.3.28  template<typename K, typename V, typename
             COMPARATOR = std::less<K>> virtual void
             decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR
             >::wait (    ) throw ( decaf::lang::exceptions::RuntimeException,
             decaf::lang::exceptions::IllegalMonitorStateException,
             decaf::lang::exceptions::InterruptedException ) [inline, virtual]
```

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling.

Exceptions

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the **Synchronizable** (p. 2930) Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2937).

```
6.189.3.29  template<typename K, typename V, typename COMPARATOR =
             std::less<K>> virtual void decaf::util::concurrent::ConcurrentStlMap<
             K, V, COMPARATOR >::wait ( long long millisecs, int
             nanos ) throw ( decaf::lang::exceptions::RuntimeException,
             decaf::lang::exceptions::IllegalArgumentException,
             decaf::lang::exceptions::IllegalMonitorStateException,
             decaf::lang::exceptions::InterruptedException ) [inline, virtual]
```

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

Parameters

millisecs the time in milliseconds to wait, or WAIT_INFINITE

nanos additional time in nanoseconds with a range of 0-999999

Exceptions

IllegalArgumentException if an error occurs or the nanos argument is not in the range of [0-999999]

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the **Synchronizable** (p. 2930) Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2940).

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**ConcurrentStlMap.h**

6.190 decaf::util::concurrent::locks::Condition Class Reference

Condition (p. 1040) factors out the **Mutex** (p. 2239) monitor methods (wait, notify and notifyAll) into distinct objects to give the effect of having multiple wait-sets per object, by combining them with the use of arbitrary **Lock** (p. 1898) implementations.

```
#include <src/main/decaf/util/concurrent/locks/Condition.h>
```

Public Member Functions

- virtual ~**Condition** ()
- virtual void **await** ()=0 throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::IllegalMonitorStateException)
Causes the current thread to wait until it is signaled or interrupted.
- virtual void **awaitUninterruptibly** ()=0 throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)
Causes the current thread to wait until it is signalled.
- virtual long long **awaitNanos** (long long nanosTimeout)=0 throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::IllegalMonitorStateException)
Causes the current thread to wait until it is signaled or interrupted, or the specified waiting time elapses.
- virtual bool **await** (long long time, const **TimeUnit** &unit)=0 throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::IllegalMonitorStateException)

Causes the current thread to wait until it is signaled or interrupted, or the specified waiting time elapses.

- virtual bool **awaitUntil** (const **Date** &deadline)=0 throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::IllegalMonitorStateException)
- virtual void **signal** ()=0 throw (decaf::lang::exceptions::RuntimeException)

Wakes up one waiting thread.

- virtual void **signalAll** ()=0 throw (decaf::lang::exceptions::RuntimeException)

Wakes up all waiting threads.

6.190.1 Detailed Description

Condition (p. 1040) factors out the **Mutex** (p. 2239) monitor methods (wait, notify and notifyAll) into distinct objects to give the effect of having multiple wait-sets per object, by combining them with the use of arbitrary **Lock** (p. 1898) implementations. Where a **Lock** (p. 1898) replaces the use of synchronized statements, a **Condition** (p. 1040) replaces the use of the Object monitor methods.

Conditions (also known as condition queues or condition variables) provide a means for one thread to suspend execution (to "wait") until notified by another thread that some state condition may now be true. Because access to this shared state information occurs in different threads, it must be protected, so a lock of some form is associated with the condition. The key property that waiting for a condition provides is that it atomically releases the associated lock and suspends the current thread.

A **Condition** (p. 1040) instance is intrinsically bound to a lock. To obtain a **Condition** (p. 1040) instance for a particular **Lock** (p. 1898) instance use its newCondition() method.

As an example, suppose we have a bounded buffer which supports put and take methods. If a take is attempted on an empty buffer, then the thread will block until an item becomes available; if a put is attempted on a full buffer, then the thread will block until a space becomes available. We would like to keep waiting put threads and take threads in separate wait-sets so that we can use the optimization of only notifying a single thread at a time when items or spaces become available in the buffer. This can be achieved using two **Condition** (p. 1040) instances.

```
class BoundedBuffer { Lock* lock = new ReentrantLock(); Condition* notFull = lock->newCondition(); Condition* notEmpty = lock->newCondition();
```

```
Object* items = new Object[100]; int putptr, takeptr, count;
```

```
public void put( Object* x ) throw( InterruptedException ) { lock->lock(); try { while( count == 100 ) notFull->await() (p. 1042); items[putptr] = x; if (++putptr == 100) putptr = 0; ++count; notEmpty->signal() (p. 1046); } catch(...) { lock->unlock(); } }
```

```
public Object take() throw( InterruptedException ) { lock->lock(); try { while(count == 0) notEmpty->await() (p. 1042); Object x = items[takeptr]; if (++takeptr == 100) takeptr = 0; --count; notFull->signal() (p. 1046); return x; } catch(...) { lock->unlock(); } }
```

(The ArrayBlockingQueue class provides this functionality, so there is no reason to implement this sample usage class.)

Implementation Considerations

When waiting upon a **Condition** (p. 1040), a "spurious wakeup" is permitted to occur, in general, as a concession to the underlying platform semantics. This has little practical impact on most

application programs as a **Condition** (p. 1040) should always be waited upon in a loop, testing the state predicate that is being waited for. An implementation is free to remove the possibility of spurious wakeups but it is recommended that applications programmers always assume that they can occur and so always wait in a loop.

The three forms of condition waiting (interruptible, non-interruptible, and timed) may differ in their ease of implementation on some platforms and in their performance characteristics. In particular, it may be difficult to provide these features and maintain specific semantics such as ordering guarantees. Further, the ability to interrupt the actual suspension of the thread may not always be feasible to implement on all platforms.

Consequently, an implementation is not required to define exactly the same guarantees or semantics for all three forms of waiting, nor is it required to support interruption of the actual suspension of the thread.

An implementation is required to clearly document the semantics and guarantees provided by each of the waiting methods, and when an implementation does support interruption of thread suspension then it must obey the interruption semantics as defined in this interface.

As interruption generally implies cancellation, and checks for interruption are often infrequent, an implementation can favor responding to an interrupt over normal method return. This is true even if it can be shown that the interrupt occurred after another action may have unblocked the thread. An implementation should document this behavior.

Since

1.0

6.190.2 Constructor & Destructor Documentation

6.190.2.1 `virtual decaf::util::concurrent::locks::Condition::~~Condition ()`
[inline, virtual]

6.190.3 Member Function Documentation

6.190.3.1 `virtual void decaf::util::concurrent::locks::Condition::await () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::IllegalMonitorStateException)` [pure virtual]

Causes the current thread to wait until it is signaled or interrupted.

The lock associated with this **Condition** (p. 1040) is atomically released and the current thread becomes disabled for thread scheduling purposes and lies dormant until one of four things happens:

* Some other thread invokes the **signal()** (p. 1046) method for this **Condition** (p. 1040) and the current thread happens to be chosen as the thread to be awakened; or * Some other thread invokes the **signalAll()** (p. 1046) method for this **Condition** (p. 1040); or * Some other thread interrupts the current thread, and interruption of thread suspension is supported; or * A "spurious wakeup" occurs.

In all cases, before this method can return the current thread must re-acquire the lock associated with this condition. When the thread returns it is guaranteed to hold this lock.

If the current thread:

* has its interrupted status set on entry to this method; or * is interrupted while waiting and interruption of thread suspension is supported,

then `InterruptedException` is thrown and the current thread's interrupted status is cleared. It is not specified, in the first case, whether or not the test for interruption occurs before the lock is released.

Implementation Considerations

The current thread is assumed to hold the lock associated with this **Condition** (p. 1040) when this method is called. It is up to the implementation to determine if this is the case and if not, how to respond. Typically, an exception will be thrown (such as `IllegalMonitorStateException`) and the implementation must document that fact.

An implementation can favor responding to an interrupt over normal method return in response to a signal. In that case the implementation must ensure that the signal is redirected to another waiting thread, if there is one.

Exceptions

RuntimeException if an unexpected error occurs while trying to wait on the **Condition** (p. 1040).

InterruptedException if the current thread is interrupted (and interruption of thread suspension is supported)

IllegalMonitorStateException if the caller is not the lock owner.

```
6.190.3.2  virtual bool decaf::util::concurrent::locks::Condition::await
            ( long long time, const TimeUnit & unit )
            throw ( decaf::lang::exceptions::RuntimeException,
                    decaf::lang::exceptions::InterruptedException,
                    decaf::lang::exceptions::IllegalMonitorStateException ) [pure virtual]
```

Causes the current thread to wait until it is signaled or interrupted, or the specified waiting time elapses.

This method is behaviorally equivalent to:

```
awaitNanos(unit.toNanos(time)) > 0
```

Parameters

time - the maximum time to wait

unit - the time unit of the time argument

Returns

false if the waiting time detectably elapsed before return from the method, else true

Exceptions

RuntimeException if an unexpected error occurs while trying to wait on the **Condition** (p. 1040).

InterruptedException if the current thread is interrupted (and interruption of thread suspension is supported)

IllegalMonitorStateException if the caller is not the lock owner.

```

6.190.3.3 virtual long long decaf::util::concurrent::locks::Condition::awaitNanos
( long long nanosTimeout ) throw ( de-
caf::lang::exceptions::RuntimeException,
decaf::lang::exceptions::InterruptedException, de-
caf::lang::exceptions::IllegalMonitorStateException ) [pure
virtual]

```

Causes the current thread to wait until it is signaled or interrupted, or the specified waiting time elapses.

The lock associated with this condition is atomically released and the current thread becomes disabled for thread scheduling purposes and lies dormant until one of five things happens:

- * Some other thread invokes the **signal()** (p. 1046) method for this **Condition** (p. 1040) and the current thread happens to be chosen as the thread to be awakened; or
- * Some other thread invokes the **signalAll()** (p. 1046) method for this **Condition** (p. 1040); or
- * Some other thread interrupts the current thread, and interruption of thread suspension is supported; or
- * The specified waiting time elapses; or
- * A "spurious wakeup" occurs.

In all cases, before this method can return the current thread must re-acquire the lock associated with this condition. When the thread returns it is guaranteed to hold this lock.

If the current thread:

- * has its interrupted status set on entry to this method; or
- * is interrupted while waiting and interruption of thread suspension is supported,

then **InterruptedException** is thrown and the current thread's interrupted status is cleared. It is not specified, in the first case, whether or not the test for interruption occurs before the lock is released.

The method returns an estimate of the number of nanoseconds remaining to wait given the supplied *nanosTimeout* value upon return, or a value less than or equal to zero if it timed out. This value can be used to determine whether and how long to re-wait in cases where the wait returns but an awaited condition still does not hold. Typical uses of this method take the following form:

```

synchronized boolean aMethod( long timeout, const TimeUnit& unit ) { long nanosTimeout =
unit.toNanos(timeout); while (!conditionBeingWaitedFor) { if (nanosTimeout > 0) nanosTimeout
= theCondition->awaitNanos(nanosTimeout); else return false; } // ... }

```

Design note: This method requires a nanosecond argument so as to avoid truncation errors in reporting remaining times. Such precision loss would make it difficult for programmers to ensure that total waiting times are not systematically shorter than specified when re-waits occur.

Implementation Considerations

The current thread is assumed to hold the lock associated with this **Condition** (p. 1040) when this method is called. It is up to the implementation to determine if this is the case and if not, how to respond. Typically, an exception will be thrown (such as **IllegalMonitorStateException**) and the implementation must document that fact.

An implementation can favor responding to an interrupt over normal method return in response to a signal, or over indicating the elapse of the specified waiting time. In either case the implementation must ensure that the signal is redirected to another waiting thread, if there is one.

Parameters

nanosTimeout - the maximum time to wait, in nanoseconds

Returns

an estimate of the nanosTimeout value minus the time spent waiting upon return from this method. A positive value may be used as the argument to a subsequent call to this method to finish waiting out the desired time. A value less than or equal to zero indicates that no time remains.

Exceptions

RuntimeException if an unexpected error occurs while trying to wait on the **Condition** (p. 1040).

InterruptedException if the current thread is interrupted (and interruption of thread suspension is supported)

IllegalMonitorStateException if the caller is not the lock owner.

6.190.3.4 virtual void decaf::util::concurrent::locks::Condition::awaitUninterruptibly () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException) [pure virtual]

Causes the current thread to wait until it is signalled.

The lock associated with this condition is atomically released and the current thread becomes disabled for thread scheduling purposes and lies dormant until one of three things happens:

* Some other thread invokes the **signal()** (p. 1046) method for this **Condition** (p. 1040) and the current thread happens to be chosen as the thread to be awakened; or * Some other thread invokes the **signalAll()** (p. 1046) method for this **Condition** (p. 1040); or * A "spurious wakeup" occurs.

In all cases, before this method can return the current thread must re-acquire the lock associated with this condition. When the thread returns it is guaranteed to hold this lock.

If the current thread's interrupted status is set when it enters this method, or it is interrupted while waiting, it will continue to wait until signalled. When it finally returns from this method its interrupted status will still be set.

Implementation Considerations

The current thread is assumed to hold the lock associated with this **Condition** (p. 1040) when this method is called. It is up to the implementation to determine if this is the case and if not, how to respond. Typically, an exception will be thrown (such as **IllegalMonitorStateException**) and the implementation must document that fact.

Exceptions

RuntimeException if an unexpected error occurs while trying to wait on the **Condition** (p. 1040).

IllegalMonitorStateException if the caller is not the lock owner.

6.190.3.5 virtual bool decaf::util::concurrent::locks::Condition::awaitUntil (const Date & *deadline*) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::IllegalMonitorStateException) [pure virtual]

6.190.3.6 virtual void decaf::util::concurrent::locks::Condition::signal () throw (decaf::lang::exceptions::RuntimeException) [pure virtual]

Wakes up one waiting thread.

If any threads are waiting on this condition then one is selected for waking up. That thread must then re-acquire the lock before returning from await.

Exceptions

RuntimeException if an unexpected error occurs while trying to wait on the **Condition** (p.1040).

6.190.3.7 virtual void decaf::util::concurrent::locks::Condition::signalAll () throw (decaf::lang::exceptions::RuntimeException) [pure virtual]

Wakes up all waiting threads.

If any threads are waiting on this condition then they are all woken up. Each thread must re-acquire the lock before it can return from await.

Exceptions

RuntimeException if an unexpected error occurs while trying to wait on the **Condition** (p.1040).

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/locks/**Condition.h**

6.191 decaf::util::concurrent::ConditionHandle Class Reference

```
#include <src/main/decaf/internal/util/concurrent/unix/ConditionHandle.h>
```

Public Member Functions

- ConditionHandle ()
- ~ConditionHandle ()
- ConditionHandle ()
- ~ConditionHandle ()

Data Fields

- `pthread_cond_t condition`
- `MutexHandle * mutex`
- `HANDLE semaphore`
- `CRITICAL_SECTION criticalSection`
- volatile unsigned int `numWaiting`
- volatile unsigned int `numWake`
- volatile unsigned int `generation`

6.191.1 Constructor & Destructor Documentation

- 6.191.1.1 `decaf::util::concurrent::ConditionHandle::ConditionHandle ()`
[inline]
- 6.191.1.2 `decaf::util::concurrent::ConditionHandle::~~ConditionHandle ()`
[inline]
- 6.191.1.3 `decaf::util::concurrent::ConditionHandle::ConditionHandle ()`
[inline]
- 6.191.1.4 `decaf::util::concurrent::ConditionHandle::~~ConditionHandle ()`
[inline]

6.191.2 Field Documentation

- 6.191.2.1 `pthread_cond_t decaf::util::concurrent::ConditionHandle::condition`
- 6.191.2.2 `CRITICAL_SECTION decaf::util::concurrent::ConditionHandle::criticalSection`
- 6.191.2.3 `volatile unsigned int decaf::util::concurrent::ConditionHandle::generation`
- 6.191.2.4 `MutexHandle * decaf::util::concurrent::ConditionHandle::mutex`
- 6.191.2.5 `volatile unsigned int decaf::util::concurrent::ConditionHandle::numWaiting`
- 6.191.2.6 `volatile unsigned int decaf::util::concurrent::ConditionHandle::numWake`
- 6.191.2.7 `HANDLE decaf::util::concurrent::ConditionHandle::semaphore`

The documentation for this class was generated from the following files:

- `src/main/decaf/internal/util/concurrent/unix/ConditionHandle.h`
- `src/main/decaf/internal/util/concurrent/windows/ConditionHandle.h`

6.192 decaf::internal::util::concurrent::ConditionImpl Class Reference

```
#include <src/main/decaf/internal/util/concurrent/ConditionImpl.h>
```


Static Public Member Functions

- static **decaf::util::concurrent::ConditionHandle** * **create** (**decaf::util::concurrent::MutexHandle** *mutex)
Creates the Condition object and attaches it to the given MutexHandle.
- static void **destroy** (**decaf::util::concurrent::ConditionHandle** *handle)
Destroy a previously create Condition instance.
- static void **wait** (**decaf::util::concurrent::ConditionHandle** *condition)
Waits for the condition to be signaled.
- static void **wait** (**decaf::util::concurrent::ConditionHandle** *condition, long long mills, long long nanos)
Waits for the condition to be signaled or for the time specified to ellapse.
- static void **notify** (**decaf::util::concurrent::ConditionHandle** *condition)
Signals one Thread that is waiting on this condition to wake up.
- static void **notifyAll** (**decaf::util::concurrent::ConditionHandle** *condition)
Signals all Threads that is waiting on this condition to wake up.

6.192.1 Member Function Documentation

6.192.1.1 static **decaf::util::concurrent::ConditionHandle***
decaf::internal::util::concurrent::ConditionImpl::create (
decaf::util::concurrent::MutexHandle * *mutex*) [static]

Creates the Condition object and attaches it to the given MutexHandle.

Parameters

mutex the Mutex handle that this Condition is attached to.

Returns

a newly constructed Condition handle that is attached to the given handle.

6.192.1.2 static void **decaf::internal::util::concurrent::ConditionImpl::destroy** (
decaf::util::concurrent::ConditionHandle * *handle*) [static]

Destroy a previously create Condition instance.

Parameters

handle The Condition handle to be destroyed.

6.192.1.3 `static void decaf::internal::util::concurrent::ConditionImpl::notify (decaf::util::concurrent::ConditionHandle * condition) [static]`

Signals one Thread that is waiting on this condition to wake up.

Parameters

condition the handle to the condition to wait on.

6.192.1.4 `static void decaf::internal::util::concurrent::ConditionImpl::notifyAll (decaf::util::concurrent::ConditionHandle * condition) [static]`

Signals all Threads that is waiting on this condition to wake up.

Parameters

condition the handle to the condition to wait on.

6.192.1.5 `static void decaf::internal::util::concurrent::ConditionImpl::wait (decaf::util::concurrent::ConditionHandle * condition, long long mills, long long nanos) [static]`

Waits for the condition to be signaled or for the time specified to ellapse.

Parameters

condition the handle to the condition to wait on.

mills the time in milliseconds to wait for the condition to be signaled.

nanos additional time in nanoseconds to wait for the thread to be signaled.

6.192.1.6 `static void decaf::internal::util::concurrent::ConditionImpl::wait (decaf::util::concurrent::ConditionHandle * condition) [static]`

Waits for the condition to be signaled.

Parameters

condition the handle to the condition to wait on.

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/util/concurrent/ConditionImpl.h`

6.193 decaf::net::ConnectException Class Reference

```
#include <src/main/decaf/net/ConnectException.h>
```

Inheritance diagram for decaf::net::ConnectException:

Public Member Functions

- **ConnectException** () throw ()
Default Constructor.
- **ConnectException** (const Exception &ex) throw ()
Conversion Constructor from some other Exception.
- **ConnectException** (const **ConnectException** &ex) throw ()
Copy Constructor.
- **ConnectException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **ConnectException** (const std::exception *cause) throw ()
Constructor.
- **ConnectException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **ConnectException** * **clone** () const
Clones this exception.
- virtual ~**ConnectException** () throw ()

6.193.1 Constructor & Destructor Documentation

6.193.1.1 decaf::net::ConnectException::ConnectException () throw () [inline]

Default Constructor.

6.193.1.2 decaf::net::ConnectException::ConnectException (const Exception &ex) throw () [inline]

Conversion Constructor from some other Exception.

Parameters

ex An exception that should become this type of Exception

6.193.1.3 decaf::net::ConnectException::ConnectException (const ConnectException & ex) throw () [inline]

Copy Constructor.

Parameters

ex An exception that should become this type of Exception

6.193.1.4 `decaf::net::ConnectException::ConnectException (const char * file,
const int lineNumber, const std::exception * cause, const char * msg,
...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.193.1.5 `decaf::net::ConnectException::ConnectException (const std::exception *
cause) throw () [inline]`

Constructor.

Parameters

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.193.1.6 `decaf::net::ConnectException::ConnectException (const char * file,
const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.193.1.7 `virtual decaf::net::ConnectException::~~ConnectException () throw ()
[inline, virtual]`

6.193.2 Member Function Documentation

6.193.2.1 `virtual ConnectException* decaf::net::ConnectException::clone ()
const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::net::SocketException** (p. 2794).

The documentation for this class was generated from the following file:

- src/main/decaf/net/**ConnectException.h**

6.194 cms::Connection Class Reference

The client's connection to its provider.

```
#include <src/main/cms/Connection.h>
```

Inheritance diagram for cms::Connection:

Public Member Functions

- virtual **~Connection** ()
- virtual void **close** ()=0 throw (CMSEException)
Closes this connection as well as any Sessions created from it (and those Sessions' consumers and producers).
- virtual const **ConnectionMetaData** * **getMetaData** () const =0 throw (CMSEException)
Gets the metadata for this connection.
- virtual **Session** * **createSession** ()=0 throw (CMSEException)
Creates an AUTO_ACKNOWLEDGE Session (p. 2663).
- virtual **Session** * **createSession** (Session::AcknowledgeMode ackMode)=0 throw (CMSEException)
Creates a new Session (p. 2663) to work for this Connection (p. 1052) using the specified acknowledgment mode.
- virtual std::string **getClientID** () const =0
Get the Client Id for this session.
- virtual **ExceptionListener** * **getExceptionListener** () const =0
Gets the registered Exception Listener for this connection.
- virtual void **setExceptionListener** (ExceptionListener *listener)=0
Sets the registered Exception Listener for this connection.

6.194.1 Detailed Description

The client's connection to its provider. Connections support concurrent use.

A connection serves several purposes:

- It encapsulates an open connection with a JMS provider. It typically represents an open TCP/IP socket between a client and the service provider software.
- Its creation is where client authentication takes place.
- It can specify a unique client identifier.
- It provides a **ConnectionMetaData** (p. 1154) object.
- It supports an optional **ExceptionListener** (p. 1483) object.

Because the creation of a connection involves setting up authentication and communication, a connection is a relatively heavy-weight object. Most clients will do all their messaging with a single connection. Other more advanced applications may use several connections. The CMS API does not architect a reason for using multiple connections; however, there may be operational reasons for doing so.

A CMS client typically creates a connection, one or more sessions, and a number of message producers and consumers. When a connection is created, it is in stopped mode. That means that no messages are being delivered.

It is typical to leave the connection in stopped mode until setup is complete (that is, until all message consumers have been created). At that point, the client calls the connection's start method, and messages begin arriving at the connection's consumers. This setup convention minimizes any client confusion that may result from asynchronous message delivery while the client is still in the process of setting itself up.

A connection can be started immediately, and the setup can be done afterwards. Clients that do this must be prepared to handle asynchronous message delivery while they are still in the process of setting up.

A message producer can send messages while a connection is stopped.

Since

1.0

6.194.2 Constructor & Destructor Documentation

6.194.2.1 `virtual cms::Connection::~~Connection () [inline, virtual]`

6.194.3 Member Function Documentation

6.194.3.1 `virtual void cms::Connection::close () throw (CMSException) [pure virtual]`

Closes this connection as well as any Sessions created from it (and those Sessions' consumers and producers).

Exceptions

CMSException (p. 960)

Implements **cms::Closeable** (p. 952).

6.194.3.2 `virtual Session* cms::Connection::createSession (Session::AcknowledgeMode ackMode) throw (CMSEException) [pure virtual]`

Creates a new **Session** (p. 2663) to work for this **Connection** (p. 1052) using the specified acknowledgment mode.

Parameters

ackMode the Acknowledgment Mode to use.

Exceptions

CMSEException (p. 960)

6.194.3.3 `virtual Session* cms::Connection::createSession () throw (CMSEException) [pure virtual]`

Creates an AUTO_ACKNOWLEDGE **Session** (p. 2663).

Exceptions

CMSEException (p. 960)

6.194.3.4 `virtual std::string cms::Connection::getClientID () const [pure virtual]`

Get the Client Id for this session.

Returns

Client Id String

6.194.3.5 `virtual ExceptionListener* cms::Connection::getExceptionListener () const [pure virtual]`

Gets the registered Exception Listener for this connection.

Returns

pointer to an exception listener or NULL

6.194.3.6 `virtual const ConnectionMetaData* cms::Connection::getMetaData () const throw (CMSEException) [pure virtual]`

Gets the metadata for this connection.

Returns

the connection MetaData pointer (caller does not own it).

Exceptions

CMSEException (p. 960) if the provider fails to get the connection metadata for this connection.

See also

ConnectionMetaData (p. 1154)

Since

2.0

6.194.3.7 `virtual void cms::Connection::setExceptionListener (ExceptionListener * listener)` [pure virtual]

Sets the registered Exception Listener for this connection.

Parameters

listener pointer to and ExceptionListener (p. 1483)

The documentation for this class was generated from the following file:

- `src/main/cms/Connection.h`

6.195 activemq::commands::ConnectionControl Class Reference

```
#include <src/main/activemq/commands/ConnectionControl.h>
```

Inheritance diagram for activemq::commands::ConnectionControl:

Public Member Functions

- **ConnectionControl** ()
- virtual **~ConnectionControl** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **ConnectionControl** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)

Copy the contents of the passed object into this object's members, overwriting any existing data.

- virtual std::string **toString** () const

*Returns a string containing the information for this **DataStructure** (p. 1372) such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** *value) const

*Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent.*

- virtual bool **isClose** () const
- virtual void **setClose** (bool **close**)
- virtual bool **isExit** () const
- virtual void **setExit** (bool **exit**)
- virtual bool **isFaultTolerant** () const
- virtual void **setFaultTolerant** (bool **faultTolerant**)
- virtual bool **isResume** () const
- virtual void **setResume** (bool **resume**)
- virtual bool **isSuspend** () const
- virtual void **setSuspend** (bool **suspend**)
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor *visitor) throw (exceptions::ActiveMQException)

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_CONNECTIONCONTROL** = 18

Protected Member Functions

- **ConnectionControl** (const **ConnectionControl** &)
- **ConnectionControl** & **operator=** (const **ConnectionControl** &)

Protected Attributes

- bool **close**
- bool **exit**
- bool **faultTolerant**
- bool **resume**
- bool **suspend**

6.195.1 Constructor & Destructor Documentation

6.195.1.1 `activemq::commands::ConnectionControl::ConnectionControl (const ConnectionControl &)` [inline, protected]

6.195.1.2 `activemq::commands::ConnectionControl::ConnectionControl ()`

6.195.1.3 `virtual activemq::commands::ConnectionControl::~~ConnectionControl ()` [virtual]

6.195.2 Member Function Documentation

6.195.2.1 `virtual ConnectionControl* activemq::commands::ConnectionControl::cloneDataStructure () const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1373).

6.195.2.2 `virtual void activemq::commands::ConnectionControl::copyDataStructure (const DataStructure * src)` [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

Reimplemented from `activemq::commands::BaseCommand` (p. 598).

6.195.2.3 `virtual bool activemq::commands::ConnectionControl::equals (const DataStructure * value) const` [virtual]

Compares the `DataStructure` (p. 1372) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 598).

6.195.2.4 `virtual unsigned char activemq::commands::ConnectionControl::getDataStructureType () const` [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataSet** (p. 1372) type copy.

Implements **activemq::commands::DataSet** (p. 1375).

6.195.2.5 virtual bool activemq::commands::ConnectionControl::isClose () const [virtual]

6.195.2.6 virtual bool activemq::commands::ConnectionControl::isExit () const [virtual]

6.195.2.7 virtual bool activemq::commands::ConnectionControl::isFaultTolerant () const [virtual]

6.195.2.8 virtual bool activemq::commands::ConnectionControl::isResume () const [virtual]

6.195.2.9 virtual bool activemq::commands::ConnectionControl::isSuspend () const [virtual]

6.195.2.10 ConnectionControl& activemq::commands::ConnectionControl::operator= (const ConnectionControl &) [inline, protected]

6.195.2.11 virtual void activemq::commands::ConnectionControl::setClose (bool *close*) [virtual]

6.195.2.12 virtual void activemq::commands::ConnectionControl::setExit (bool *exit*) [virtual]

6.195.2.13 virtual void activemq::commands::ConnectionControl::setFaultTolerant (bool *faultTolerant*) [virtual]

6.195.2.14 virtual void activemq::commands::ConnectionControl::setResume (bool *resume*) [virtual]

6.195.2.15 virtual void activemq::commands::ConnectionControl::setSuspend (bool *suspend*) [virtual]

6.195.2.16 virtual std::string activemq::commands::ConnectionControl::toString () const [virtual]

Returns a string containing the information for this **DataSet** (p. 1372) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 602).

```

6.195.2.17 virtual Pointer<Command> activemq::commands::ConnectionControl::visit (
    activemq::state::CommandVisitor * visitor ) throw (
    exceptions::ActiveMQException ) [virtual]

```

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

Returns

a **Response** (p. 2611) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 995).

6.195.3 Field Documentation

- 6.195.3.1** bool **activemq::commands::ConnectionControl::close** [protected]
- 6.195.3.2** bool **activemq::commands::ConnectionControl::exit** [protected]
- 6.195.3.3** bool **activemq::commands::ConnectionControl::faultTolerant** [protected]
- 6.195.3.4** const unsigned char **activemq::commands::ConnectionControl::ID_ - CONNECTIONCONTROL = 18** [static]
- 6.195.3.5** bool **activemq::commands::ConnectionControl::resume** [protected]
- 6.195.3.6** bool **activemq::commands::ConnectionControl::suspend** [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ConnectionControl.h`

6.196 activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller Class Reference

Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1059).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ConnectionControlMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller**:

Public Member Functions

- **ConnectionControlMarshaller** ()
- virtual **~ConnectionControlMarshaller** ()
- virtual **commands::DataStructure * createObject** () const

Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)

Un-marshall an object instance from the data input stream.

- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshall an object instance from the data input stream.

- virtual void **looseMarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.196.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p.1059). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.196.2 Constructor & Destructor Documentation

6.196.2.1 `activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller::ConnectionControlMarshaller()` [inline]

6.196.2.2 `virtual activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller::~~ConnectionControlMarshaller()` [inline, virtual]

6.196.3 Member Function Documentation

6.196.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.196.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.196.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 604).

6.196.3.4 virtual void activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 605).

6.196.3.5 virtual int activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 606).

6.196.3.6 virtual void activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 608).

```
6.196.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 609).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/ConnectionControlMarshaller.h

6.197 **activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller** Class Reference

Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1063).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ConnectionControlMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller**:

Public Member Functions

- **ConnectionControlMarshaller** ()
- virtual **~ConnectionControlMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.197.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p.1063). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.197.2 Constructor & Destructor Documentation

6.197.2.1 `activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller::ConnectionControlMarshaller () [inline]`

6.197.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller::~~ConnectionControlMarshaller () [inline, virtual]`

6.197.3 Member Function Documentation

6.197.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.197.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.197.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 611).

6.197.3.4 virtual void activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 612).

6.197.3.5 virtual int activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 613).

6.197.3.6 virtual void activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 614).

```
6.197.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 615).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/ConnectionControlMarshaller.h`

6.198 `activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller` Class Reference

Marshaling code for Open Wire Format for `ConnectionControlMarshaller` (p. 1067).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ConnectionControlMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller`:

Public Member Functions

- **ConnectionControlMarshaller** ()
- virtual **~ConnectionControlMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.198.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p.1067). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.198.2 Constructor & Destructor Documentation

6.198.2.1 `activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller::ConnectionControlMarshaller () [inline]`

6.198.2.2 `virtual activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller::~ConnectionControlMarshaller () [inline, virtual]`

6.198.3 Member Function Documentation

6.198.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.198.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.198.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 618).

6.198.3.4 virtual void activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 619).

6.198.3.5 virtual int activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 620).

6.198.3.6 virtual void activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 621).

```
6.198.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 622).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/ConnectionControlMarshaller.h

6.199 activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller Class Reference

Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1071).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ConnectionControlMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller**:

Public Member Functions

- **ConnectionControlMarshaller** ()
- virtual **~ConnectionControlMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.199.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p.1071). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.199.2 Constructor & Destructor Documentation

6.199.2.1 `activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller::ConnectionControlMarshaller () [inline]`

6.199.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller::~~ConnectionControlMarshaller () [inline, virtual]`

6.199.3 Member Function Documentation

6.199.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.199.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.199.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 624).

6.199.3.4 virtual void activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 625).

6.199.3.5 virtual int activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 626).

6.199.3.6 virtual void activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 628).

```
6.199.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 629).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ConnectionControlMarshaller.h`

6.200 `activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller` Class Reference

Marshaling code for Open Wire Format for `ConnectionControlMarshaller` (p. 1075).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ConnectionControlMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller`:

Public Member Functions

- **ConnectionControlMarshaller** ()
- virtual **~ConnectionControlMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.200.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p.1075). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.200.2 Constructor & Destructor Documentation

6.200.2.1 `activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller::ConnectionControlMarshaller () [inline]`

6.200.2.2 `virtual activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller::~~ConnectionControlMarshaller () [inline, virtual]`

6.200.3 Member Function Documentation

6.200.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.200.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.200.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 631).

6.200.3.4 virtual void activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 632).

6.200.3.5 virtual int activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 633).

6.200.3.6 virtual void activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions

- IOException* if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 634).

```
6.200.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 635).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ConnectionControlMarshaller.h`

6.201 `activemq::commands::ConnectionError` Class Reference

```
#include <src/main/activemq/commands/ConnectionError.h>
```

Inheritance diagram for `activemq::commands::ConnectionError`:

Public Member Functions

- **ConnectionError** ()
- virtual **~ConnectionError** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **ConnectionError * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1372) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **BrokerError** > & **getException** () const
- virtual **Pointer**< **BrokerError** > & **getException** ()
- virtual void **setException** (const **Pointer**< **BrokerError** > &exception)
- virtual const **Pointer**< **ConnectionId** > & **getConnectionId** () const
- virtual **Pointer**< **ConnectionId** > & **getConnectionId** ()
- virtual void **setConnectionId** (const **Pointer**< **ConnectionId** > &connectionId)
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor *visitor) throw (exceptions::ActiveMQException)
Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID _ CONNECTIONERROR** = 16

Protected Member Functions

- **ConnectionError** (const **ConnectionError** &)
- **ConnectionError** & **operator=** (const **ConnectionError** &)

Protected Attributes

- **Pointer**< **BrokerError** > exception
- **Pointer**< **ConnectionId** > connectionId

6.201.1 Constructor & Destructor Documentation

- 6.201.1.1** `activemq::commands::ConnectionError::ConnectionError (const ConnectionError &) [inline, protected]`
- 6.201.1.2** `activemq::commands::ConnectionError::ConnectionError ()`
- 6.201.1.3** `virtual activemq::commands::ConnectionError::~~ConnectionError () [virtual]`

6.201.2 Member Function Documentation

- 6.201.2.1** `virtual ConnectionError* activemq::commands::ConnectionError::cloneDataStructure () const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1373).

- 6.201.2.2** `virtual void activemq::commands::ConnectionError::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

Reimplemented from `activemq::commands::BaseCommand` (p. 598).

- 6.201.2.3** `virtual bool activemq::commands::ConnectionError::equals (const DataStructure * value) const [virtual]`

Compares the `DataStructure` (p. 1372) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 598).

- 6.201.2.4 `virtual const Pointer<ConnectionId>& activemq::commands::ConnectionError::getConnectionId () const [virtual]`
- 6.201.2.5 `virtual Pointer<ConnectionId>& activemq::commands::ConnectionError::getConnectionId () [virtual]`
- 6.201.2.6 `virtual unsigned char activemq::commands::ConnectionError::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataSet** (p. 1372) type copy.

Implements **activemq::commands::DataSet** (p. 1375).

- 6.201.2.7 `virtual Pointer<BrokerError>& activemq::commands::ConnectionError::getException () [virtual]`
- 6.201.2.8 `virtual const Pointer<BrokerError>& activemq::commands::ConnectionError::getException () const [virtual]`
- 6.201.2.9 `ConnectionError& activemq::commands::ConnectionError::operator= (const ConnectionError &) [inline, protected]`
- 6.201.2.10 `virtual void activemq::commands::ConnectionError::setConnectionId (const Pointer< ConnectionId > & connectionId) [virtual]`
- 6.201.2.11 `virtual void activemq::commands::ConnectionError::setException (const Pointer< BrokerError > & exception) [virtual]`
- 6.201.2.12 `virtual std::string activemq::commands::ConnectionError::toString () const [virtual]`

Returns a string containing the information for this **DataSet** (p. 1372) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 602).

6.201.2.13 `virtual Pointer<Command> activemq::commands::ConnectionError::visit (activemq::state::CommandVisitor * visitor) throw (exceptions::ActiveMQException) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

Returns

a **Response** (p. 2611) to the visitor being called or NULL if no response.

Implements `activemq::commands::Command` (p. 995).

6.201.3 Field Documentation

6.201.3.1 `Pointer<ConnectionId> activemq::commands::ConnectionError::connectionId [protected]`

6.201.3.2 `Pointer<BrokerError> activemq::commands::ConnectionError::exception [protected]`

6.201.3.3 `const unsigned char activemq::commands::ConnectionError::ID _ - CONNECTIONERROR = 16 [static]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ConnectionError.h`

6.202 activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller Class Reference

Marshaling code for Open Wire Format for `ConnectionErrorMarshaller` (p. 1083).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ConnectionErrorMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller`:

Public Member Functions

- `ConnectionErrorMarshaller ()`
- `virtual ~ConnectionErrorMarshaller ()`
- `virtual commands::DataStructure * createObject () const`

Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.202.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p.1083). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.202.2 Constructor & Destructor Documentation

- 6.202.2.1 **activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller::ConnectionErrorMarshaller** () [inline]
- 6.202.2.2 **virtual activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller::~~ConnectionErrorMarshaller** () [inline, virtual]

6.202.3 Member Function Documentation

- 6.202.3.1 **virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller::createObject** () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1331).

6.202.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller::getDataStructure () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1336).

6.202.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 604).

6.202.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 605).

6.202.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 606).

6.202.3.6 `virtual void activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 608).

```

6.202.3.7 virtual void activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]

```

Un-marshall an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 609).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ConnectionErrorMarshaller.h`

6.203 activemq::wireformat::openwire::marshal::v1::ConnectionErrorMa Class Reference

Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1087).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ConnectionErrorMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller**:

Public Member Functions

- **ConnectionErrorMarshaller** ()
- virtual **~ConnectionErrorMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.203.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p.1087). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.203.2 Constructor & Destructor Documentation

6.203.2.1 activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller::ConnectionErrorMarshaller () [inline]

6.203.2.2 virtual
activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller::~~ConnectionErrorMarshaller () [inline, virtual]

6.203.3 Member Function Documentation

6.203.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1331).

6.203.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller::getDataStructureType() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.203.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 611).

6.203.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 612).

6.203.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 613).

6.203.3.6 `virtual void activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 614).

6.203.3.7 `virtual void activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 615).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/ConnectionErrorMarshaller.h`

6.204 `activemq::wireformat::openwire::marshal::v4::ConnectionErrorMa` Class Reference

Marshaling code for Open Wire Format for `ConnectionErrorMarshaller` (p. 1091).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ConnectionErrorMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller`:

Public Member Functions

- `ConnectionErrorMarshaller ()`
- `virtual ~ConnectionErrorMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaler.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.204.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p.1091). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.204.2 Constructor & Destructor Documentation

6.204.2.1 **activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller::ConnectionErrorMarshaller** () [inline]

6.204.2.2 **virtual activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller::~~ConnectionErrorMarshaller** () [inline, virtual]

6.204.3 Member Function Documentation

6.204.3.1 **virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller::createObject** () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1331).

6.204.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller::getDataStructureType** () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.204.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 618).

6.204.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 619).

6.204.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 620).

```
6.204.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 621).

```
6.204.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 622).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/ConnectionErrorMarshaller.h`

6.205 activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller Class Reference

Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1095).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ConnectionErrorMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller`:

Public Member Functions

- **ConnectionErrorMarshaller** ()
- virtual **~ConnectionErrorMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

- virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.205.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p.1095). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.205.2 Constructor & Destructor Documentation

6.205.2.1 activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller::ConnectionErrorMarshaller () [inline]

6.205.2.2 virtual activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller::~~ConnectionErrorMarshaller () [inline, virtual]

6.205.3 Member Function Documentation

6.205.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1331).

6.205.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1336).

6.205.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 624).

6.205.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 625).

6.205.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 626).

```
6.205.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 628).

```
6.205.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 629).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ConnectionErrorMarshaller.h`

6.206 activemq::wireformat::openwire::marshal::v2::ConnectionErrorMa Class Reference

Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1099).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ConnectionErrorMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller`:

Public Member Functions

- **ConnectionErrorMarshaller** ()
- virtual **~ConnectionErrorMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshall an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.206.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p.1099). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.206.2 Constructor & Destructor Documentation

6.206.2.1 activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller::ConnectionErrorMarshaller () [inline]

6.206.2.2 virtual activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller::~ConnectionErrorMarshaller () [inline, virtual]

6.206.3 Member Function Documentation

6.206.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1331).

6.206.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1336).

6.206.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 631).

6.206.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 632).

6.206.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 633).

```
6.206.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 634).

```
6.206.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 635).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ConnectionErrorMarshaller.h`

6.207 cms::ConnectionFactory Class Reference

Defines the interface for a factory that creates connection objects, the **Connection** (p.1052) objects returned implement the CMS **Connection** (p. 1052) interface and hide the CMS Provider specific implementation details behind that interface.

```
#include <src/main/cms/ConnectionFactory.h>
```

Inheritance diagram for cms::ConnectionFactory:

Public Member Functions

- virtual **~ConnectionFactory** ()
- virtual **Connection * createConnection** ()=0 throw (CMSEException)
Creates a connection with the default user identity.
- virtual **cms::Connection * createConnection** (const std::string &username, const std::string &password)=0 throw (cms::CMSEException)
Creates a connection with the default specified identity.
- virtual **cms::Connection * createConnection** (const std::string &username, const std::string &password, const std::string &clientId)=0 throw (cms::CMSEException)
Creates a connection with the specified user identity.

Static Public Member Functions

- static **ConnectionFactory * createCMSConnectionFactory** (const std::string &brokerURI) throw (cms::CMSEException)
Static method that is used to create a provider specific connection factory.

6.207.1 Detailed Description

Defines the interface for a factory that creates connection objects, the **Connection** (p.1052) objects returned implement the CMS **Connection** (p. 1052) interface and hide the CMS Provider specific implementation details behind that interface. A Client creates a new **ConnectionFactory** (p.1103) either directly by instantiating the provider specific implementation of the factory or

by using the static method `createCMSConnectionFactory` which all providers are required to implement.

Since

1.0

6.207.2 Constructor & Destructor Documentation

6.207.2.1 `virtual cms::ConnectionFactory::~~ConnectionFactory () [inline, virtual]`

6.207.3 Member Function Documentation

6.207.3.1 `static ConnectionFactory* cms::ConnectionFactory::createCMSConnectionFactory (const std::string & brokerURI) throw (cms::CMSException) [static]`

Static method that is used to create a provider specific connection factory.

The provider implements this method in their library and returns an instance of a **ConnectionFactory** (p. 1103) derived object. Clients can use this method to remain abstracted from the specific CMS implementation being used.

Parameters

brokerURI The remote address to use to connect to the Provider.

Returns

A pointer to a provider specific implementation of the **ConnectionFactory** (p. 1103) interface, the caller is responsible for deleting this resource.

Exceptions

CMSException (p. 960) if an internal error occurs while creating the **ConnectionFactory** (p. 1103).

6.207.3.2 `virtual cms::Connection* cms::ConnectionFactory::createConnection (const std::string & username, const std::string & password) throw (cms::CMSException) [pure virtual]`

Creates a connection with the default specified identity.

The connection is created in stopped mode. No messages will be delivered until the **Connection.start** (p. 2831) method is explicitly called. The username and password values passed here do not change the defaults, subsequent calls to the parameterless `createConnection` will continue to use the default values that were set in the Constructor.

Parameters

username The user name used to authenticate with the Provider.

password The password used to authenticate with the Provider.

Returns

A pointer to a connection object, caller owns the pointer and is responsible for closing the connection and deleting the instance.

Exceptions

CMSEException (p. 960) if an internal error occurs while creating the **Connection** (p. 1052).

Implemented in **activemq::core::ActiveMQConnectionFactory** (p. 213).

6.207.3.3 `virtual cms::Connection* cms::ConnectionFactory::createConnection (const std::string & username, const std::string & password, const std::string & clientId) throw (cms::CMSEException) [pure virtual]`

Creates a connection with the specified user identity.

The connection is created in stopped mode. No messages will be delivered until the **Connection.start** (p. 2831) method is explicitly called. The user name and password values passed here do not change the defaults, subsequent calls to the parameterless `createConnection` will continue to use the default values that were set in the Constructor.

Parameters

username The user name used to authenticate with the Provider.

password The password used to authenticate with the Provider.

clientId The Client Id assigned to connection. If the id is the empty string ("") then a random client Id is created for this connection.

Returns

A pointer to a connection object, caller owns the pointer and is responsible for closing the connection and deleting the instance.

Exceptions

CMSEException (p. 960) if an internal error occurs while creating the **Connection** (p. 1052).

Implemented in **activemq::core::ActiveMQConnectionFactory** (p. 214).

6.207.3.4 `virtual Connection* cms::ConnectionFactory::createConnection () throw (CMSEException) [pure virtual]`

Creates a connection with the default user identity.

The connection is created in stopped mode. No messages will be delivered until the **Connection.start** (p. 2831) method is explicitly called.

Returns

A pointer to a connection object, caller owns the pointer and is responsible for closing the connection and deleting the instance.

Exceptions

CMSException (p. 960) if an internal error occurs while creating the **Connection** (p. 1052).

Implemented in **activemq::core::ActiveMQConnectionFactory** (p. 213).

The documentation for this class was generated from the following file:

- src/main/cms/**ConnectionFactory.h**

6.208 activemq::commands::ConnectionId Class Reference

```
#include <src/main/activemq/commands/ConnectionId.h>
```

Inheritance diagram for **activemq::commands::ConnectionId**:

Public Types

- typedef **decaf::lang::PointerComparator**< **ConnectionId** > **COMPARATOR**

Public Member Functions

- **ConnectionId** ()
- **ConnectionId** (const **ConnectionId** &other)
- **ConnectionId** (const **SessionId** *sessionId)
- **ConnectionId** (const **ProducerId** *producerId)
- **ConnectionId** (const **ConsumerId** *consumerId)
- virtual ~**ConnectionId** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **ConnectionId** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1372) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent.*
- virtual const std::string & **getValue** () const
- virtual std::string & **getValue** ()
- virtual void **setValue** (const std::string &value)

- virtual int **compareTo** (const **ConnectionId** &value) const
- virtual bool **equals** (const **ConnectionId** &value) const
- virtual bool **operator==** (const **ConnectionId** &value) const
- virtual bool **operator<** (const **ConnectionId** &value) const
- **ConnectionId** & **operator=** (const **ConnectionId** &other)

Static Public Attributes

- static const unsigned char **ID_CONNECTIONID** = 120

Protected Attributes

- std::string **value**

6.208.1 Member Typedef Documentation

- 6.208.1.1** `typedef decaf::lang::PointerComparator<ConnectionId>
activemq::commands::ConnectionId::COMPARATOR`

6.208.2 Constructor & Destructor Documentation

- 6.208.2.1** `activemq::commands::ConnectionId::ConnectionId ()`
- 6.208.2.2** `activemq::commands::ConnectionId::ConnectionId (const ConnectionId
& other)`
- 6.208.2.3** `activemq::commands::ConnectionId::ConnectionId (const SessionId *
sessionId)`
- 6.208.2.4** `activemq::commands::ConnectionId::ConnectionId (const ProducerId *
producerId)`
- 6.208.2.5** `activemq::commands::ConnectionId::ConnectionId (const ConsumerId *
consumerId)`
- 6.208.2.6** `virtual activemq::commands::ConnectionId::~~ConnectionId ()
[virtual]`

6.208.3 Member Function Documentation

- 6.208.3.1** `virtual ConnectionId* ac-
tivemq::commands::ConnectionId::cloneDataStructure (
) const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

6.208.3.2 `virtual int activemq::commands::ConnectionId::compareTo (const ConnectionId & value) const` [virtual]

6.208.3.3 `virtual void activemq::commands::ConnectionId::copyDataStructure (const DataStructure * src)` [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

6.208.3.4 `virtual bool activemq::commands::ConnectionId::equals (const DataStructure * value) const` [virtual]

Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

6.208.3.5 `virtual bool activemq::commands::ConnectionId::equals (const ConnectionId & value) const` [virtual]

6.208.3.6 `virtual unsigned char activemq::commands::ConnectionId::getDataStructureType () const` [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataStructure** (p. 1372) type copy.

- 6.208.3.7 `virtual const std::string& activemq::commands::ConnectionId::getValue () const [virtual]`
- 6.208.3.8 `virtual std::string& activemq::commands::ConnectionId::getValue () [virtual]`
- 6.208.3.9 `virtual bool activemq::commands::ConnectionId::operator< (const ConnectionId & value) const [virtual]`
- 6.208.3.10 `ConnectionId& activemq::commands::ConnectionId::operator= (const ConnectionId & other)`
- 6.208.3.11 `virtual bool activemq::commands::ConnectionId::operator== (const ConnectionId & value) const [virtual]`
- 6.208.3.12 `virtual void activemq::commands::ConnectionId::setValue (const std::string & value) [virtual]`
- 6.208.3.13 `virtual std::string activemq::commands::ConnectionId::toString () const [virtual]`

Returns a string containing the information for this **DataSet** (p.1372) such as its type and value of its elements.

Returns

formatted string useful for debugging.

6.208.4 Field Documentation

- 6.208.4.1 `const unsigned char activemq::commands::ConnectionId::ID_ - CONNECTIONID = 120 [static]`

Referenced by `activemq::state::CommandVisitorAdapter::processRemoveInfo()`.

- 6.208.4.2 `std::string activemq::commands::ConnectionId::value [protected]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ConnectionId.h`

6.209 activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p.1109).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ConnectionIdMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller`:

Public Member Functions

- **ConnectionIdMarshaller** ()
- virtual **~ConnectionIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.209.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p.1109). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.209.2 Constructor & Destructor Documentation

6.209.2.1 `activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller::ConnectionIdMarshaller () [inline]`

6.209.2.2 `virtual activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller::~~ConnectionIdMarshaller () [inline, virtual]`

6.209.3 Member Function Documentation

6.209.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.209.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.209.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1342).

6.209.3.4 virtual void activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1348).

6.209.3.5 virtual int activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1354).

6.209.3.6 virtual void activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1360).

6.209.3.7 virtual void **activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller::tightUnmarshal**
 (**OpenWireFormat** * *wireFormat*, **commands::DataStructure**
 * *dataStructure*, **decaf::io::DataInputStream** * *dataIn*,
utils::BooleanStream * *bs*) throw (**decaf::io::IOException**) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1366).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ConnectionIdMarshaller.h`

6.210 **activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller** Class Reference

Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1113).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ConnectionIdMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller**:

Public Member Functions

- **ConnectionIdMarshaller** ()
- virtual **~ConnectionIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.210.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p.1113). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.210.2 Constructor & Destructor Documentation

6.210.2.1 `activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller::ConnectionIdMarshaller()` [inline]

6.210.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller::~~ConnectionIdMarshaller()` [inline, virtual]

6.210.3 Member Function Documentation

6.210.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.210.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.210.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller::looseMarshal(OpenWireFormat* wireFormat, commands::DataStructure* dataStructure, decaf::io::DataOutputStream* dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1342).

6.210.3.4 virtual void activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1348).

6.210.3.5 virtual int activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1354).

6.210.3.6 virtual void activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions

- IOException* if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1360).

```
6.210.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1366).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/ConnectionIdMarshaller.h`

6.211 activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1116).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ConnectionIdMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller**:

Public Member Functions

- **ConnectionIdMarshaller** ()
- virtual **~ConnectionIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.211.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p.1116). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.211.2 Constructor & Destructor Documentation

6.211.2.1 `activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller::ConnectionIdMarshaller()` [inline]

6.211.2.2 `virtual activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller::~~ConnectionIdMarshaller()` [inline, virtual]

6.211.3 Member Function Documentation

6.211.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.211.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.211.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut)` throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1342).

6.211.3.4 virtual void activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1348).

6.211.3.5 virtual int activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1354).

6.211.3.6 virtual void activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions

- IOException* if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1360).

```
6.211.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1366).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/ConnectionIdMarshaller.h

6.212 activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1120).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ConnectionIdMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller**:

Public Member Functions

- **ConnectionIdMarshaller** ()
- virtual **~ConnectionIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.212.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p.1120). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.212.2 Constructor & Destructor Documentation

6.212.2.1 `activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller::ConnectionIdMarshaller()` [inline]

6.212.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller::~~ConnectionIdMarshaller()` [inline, virtual]

6.212.3 Member Function Documentation

6.212.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.212.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.212.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut)` throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1342).

6.212.3.4 virtual void activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1348).

6.212.3.5 virtual int activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1354).

6.212.3.6 virtual void activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1360).

6.212.3.7 virtual void **activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller::tightUnmarshal**
 (**OpenWireFormat** * *wireFormat*, **commands::DataStructure**
 * *dataStructure*, **decaf::io::DataInputStream** * *dataIn*,
utils::BooleanStream * *bs*) throw (**decaf::io::IOException**) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1366).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ConnectionIdMarshaller.h`

6.213 **activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller** Class Reference

Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1124).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ConnectionIdMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller**:

Public Member Functions

- **ConnectionIdMarshaller** ()
- virtual **~ConnectionIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.213.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p.1124). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.213.2 Constructor & Destructor Documentation

6.213.2.1 `activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller::ConnectionIdMarshaller () [inline]`

6.213.2.2 `virtual activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller::~~ConnectionIdMarshaller () [inline, virtual]`

6.213.3 Member Function Documentation

6.213.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.213.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.213.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1342).

6.213.3.4 virtual void activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1348).

6.213.3.5 virtual int activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1354).

6.213.3.6 virtual void activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1360).

```
6.213.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1366).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/ConnectionIdMarshaller.h

6.214 activemq::commands::ConnectionInfo Class Reference

```
#include <src/main/activemq/commands/ConnectionInfo.h>
```

Inheritance diagram for **activemq::commands::ConnectionInfo**:

Public Member Functions

- **ConnectionInfo** ()
- virtual **~ConnectionInfo** ()
- virtual unsigned char **getDataStructureType** () const

Get the unique identifier that this object and its own Marshaler share.

- virtual **ConnectionInfo** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1372) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **ConnectionId** > & **getConnectionId** () const
- virtual **Pointer**< **ConnectionId** > & **getConnectionId** ()
- virtual void **setConnectionId** (const **Pointer**< **ConnectionId** > &connectionId)
- virtual const std::string & **getClientId** () const
- virtual std::string & **getClientId** ()
- virtual void **setClientId** (const std::string &clientId)
- virtual const std::string & **getPassword** () const
- virtual std::string & **getPassword** ()
- virtual void **setPassword** (const std::string &password)
- virtual const std::string & **getUserName** () const
- virtual std::string & **getUserName** ()
- virtual void **setUserName** (const std::string &userName)
- virtual const std::vector< **decaf::lang::Pointer**< **BrokerId** > > & **getBrokerPath** () const
- virtual std::vector< **decaf::lang::Pointer**< **BrokerId** > > & **getBrokerPath** ()
- virtual void **setBrokerPath** (const std::vector< **decaf::lang::Pointer**< **BrokerId** > > &brokerPath)
- virtual bool **isBrokerMasterConnector** () const
- virtual void **setBrokerMasterConnector** (bool brokerMasterConnector)
- virtual bool **isManageable** () const
- virtual void **setManageable** (bool manageable)
- virtual bool **isClientMaster** () const
- virtual void **setClientMaster** (bool clientMaster)
- virtual bool **isConnectionInfo** () const
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor *visitor) throw (exceptions::ActiveMQException)
Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_CONNECTIONINFO** = 3

Protected Member Functions

- **ConnectionInfo** (const **ConnectionInfo** &)
- **ConnectionInfo** & **operator=** (const **ConnectionInfo** &)

Protected Attributes

- **Pointer**< **ConnectionId** > **connectionId**
- std::string **clientId**
- std::string **password**
- std::string **userName**
- std::vector< **decaf::lang::Pointer**< **BrokerId** > > **brokerPath**
- bool **brokerMasterConnector**
- bool **manageable**
- bool **clientMaster**

6.214.1 Constructor & Destructor Documentation

- 6.214.1.1** **activemq::commands::ConnectionInfo::ConnectionInfo** (const **ConnectionInfo** &) [inline, protected]
- 6.214.1.2** **activemq::commands::ConnectionInfo::ConnectionInfo** ()
- 6.214.1.3** **virtual activemq::commands::ConnectionInfo::~~ConnectionInfo** () [virtual]

6.214.2 Member Function Documentation

- 6.214.2.1** **virtual ConnectionInfo* activemq::commands::ConnectionInfo::cloneDataStructure** () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1373).

- 6.214.2.2** **virtual void activemq::commands::ConnectionInfo::copyDataStructure** (const **DataStructure** * *src*) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 598).

6.214.2.3 `virtual bool activemq::commands::ConnectionInfo::equals (const DataStructure * value) const` [virtual]

Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 598).

6.214.2.4 `virtual const std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::ConnectionInfo::getBrokerPath () const` [virtual]

6.214.2.5 `virtual std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::ConnectionInfo::getBrokerPath ()` [virtual]

6.214.2.6 `virtual const std::string& activemq::commands::ConnectionInfo::getClientId () const` [virtual]

6.214.2.7 `virtual std::string& activemq::commands::ConnectionInfo::getClientId ()` [virtual]

6.214.2.8 `virtual Pointer<ConnectionId>& activemq::commands::ConnectionInfo::getConnectionId ()` [virtual]

6.214.2.9 `virtual const Pointer<ConnectionId>& activemq::commands::ConnectionInfo::getConnectionId () const` [virtual]

6.214.2.10 `virtual unsigned char activemq::commands::ConnectionInfo::getDataStructureType () const` [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataStructure** (p. 1372) type copy.

Implements **activemq::commands::DataStructure** (p. 1375).

- 6.214.2.11 `virtual const std::string& activemq::commands::ConnectionInfo::getPassword ()`
`const [virtual]`
- 6.214.2.12 `virtual std::string& activemq::commands::ConnectionInfo::getPassword () [virtual]`
- 6.214.2.13 `virtual const std::string& activemq::commands::ConnectionInfo::getUserName ()`
`const [virtual]`
- 6.214.2.14 `virtual std::string& activemq::commands::ConnectionInfo::getUserName () [virtual]`
- 6.214.2.15 `virtual bool activemq::commands::ConnectionInfo::isBrokerMasterConnector ()`
`const [virtual]`
- 6.214.2.16 `virtual bool activemq::commands::ConnectionInfo::isClientMaster ()`
`const [virtual]`
- 6.214.2.17 `virtual bool activemq::commands::ConnectionInfo::isConnectionInfo () const [inline, virtual]`

Returns

an answer of true to the `isConnectionInfo()` (p. 1132) query.

Reimplemented from `activemq::commands::BaseCommand` (p. 600).

- 6.214.2.18 `virtual bool activemq::commands::ConnectionInfo::isManageable ()
const [virtual]`
- 6.214.2.19 `ConnectionInfo& activemq::commands::ConnectionInfo::operator= (
const ConnectionInfo &) [inline, protected]`
- 6.214.2.20 `virtual void ac-
tivemq::commands::ConnectionInfo::setBrokerMasterConnector (bool
brokerMasterConnector) [virtual]`
- 6.214.2.21 `virtual void activemq::commands::ConnectionInfo::setBrokerPath (
const std::vector< decaf::lang::Pointer< BrokerId > > & brokerPath)
[virtual]`
- 6.214.2.22 `virtual void activemq::commands::ConnectionInfo::setClientId (const
std::string & clientId) [virtual]`
- 6.214.2.23 `virtual void activemq::commands::ConnectionInfo::setClientMaster (
bool clientMaster) [virtual]`
- 6.214.2.24 `virtual void activemq::commands::ConnectionInfo::setConnectionId (
const Pointer< ConnectionId > & connectionId) [virtual]`
- 6.214.2.25 `virtual void activemq::commands::ConnectionInfo::setManageable (
bool manageable) [virtual]`
- 6.214.2.26 `virtual void activemq::commands::ConnectionInfo::setPassword (const
std::string & password) [virtual]`
- 6.214.2.27 `virtual void activemq::commands::ConnectionInfo::setUserName (
const std::string & userName) [virtual]`
- 6.214.2.28 `virtual std::string activemq::commands::ConnectionInfo::toString ()
const [virtual]`

Returns a string containing the information for this **DataSet** (p.1372) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 602).

- 6.214.2.29 `virtual Pointer<Command> activemq::commands::ConnectionInfo::visit
(activemq::state::CommandVisitor * visitor) throw (
exceptions::ActiveMQException) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

Returns

a **Response** (p. 2611) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 995).

6.214.3 Field Documentation

- 6.214.3.1** `bool activemq::commands::ConnectionInfo::brokerMasterConnector` [protected]
- 6.214.3.2** `std::vector< decaf::lang::Pointer<BrokerId> >`
`activemq::commands::ConnectionInfo::brokerPath` [protected]
- 6.214.3.3** `std::string activemq::commands::ConnectionInfo::clientId` [protected]
- 6.214.3.4** `bool activemq::commands::ConnectionInfo::clientMaster` [protected]
- 6.214.3.5** `Pointer<ConnectionId> ac-`
`tivemq::commands::ConnectionInfo::connectionId`
[protected]
- 6.214.3.6** `const unsigned char activemq::commands::ConnectionInfo::ID _-`
`CONNECTIONINFO = 3` [static]
- 6.214.3.7** `bool activemq::commands::ConnectionInfo::manageable` [protected]
- 6.214.3.8** `std::string activemq::commands::ConnectionInfo::password` [protected]
- 6.214.3.9** `std::string activemq::commands::ConnectionInfo::userName` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ConnectionInfo.h`

6.215 **activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller** Class Reference

Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1134).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ConnectionInfoMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller`:

Public Member Functions

- `ConnectionInfoMarshaller ()`
- `virtual ~ConnectionInfoMarshaller ()`
- `virtual commands::DataStructure * createObject () const`

Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.215.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1134). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.215.2 Constructor & Destructor Documentation

6.215.2.1 `activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller::ConnectionInfoM`
`() [inline]`

6.215.2.2 `virtual`
`activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller::~~ConnectionInfo`
`() [inline, virtual]`

6.215.3 Member Function Documentation

6.215.3.1 `virtual commands::DataStructure* ac-`
`tivemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller::createObject`
`() const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.215.3.2 `virtual unsigned char ac-`
`tivemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller::getDataStructureT`
`() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.215.3.3 `virtual void ac-`
`tivemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller::looseMarshal`
`(OpenWireFormat * wireFormat, commands::DataStructure *`
`dataStructure, decaf::io::DataOutputStream * dataOut) throw (`
`decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 604).

6.215.3.4 virtual void activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 605).

6.215.3.5 virtual int activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 606).

6.215.3.6 virtual void activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 608).

```
6.215.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 609).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ConnectionInfoMarshaller.h`

6.216 `activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller` Class Reference

Marshaling code for Open Wire Format for `ConnectionInfoMarshaller` (p. 1138).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ConnectionInfoMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller`:

Public Member Functions

- **ConnectionInfoMarshaller** ()
- virtual **~ConnectionInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.216.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1138). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.216.2 Constructor & Destructor Documentation

6.216.2.1 `activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller::ConnectionInfoM
() [inline]`

6.216.2.2 `virtual
activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller::~~ConnectionInfo
() [inline, virtual]`

6.216.3 Member Function Documentation

6.216.3.1 `virtual commands::DataStructure* ac-
tivemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller::createObject
() const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.216.3.2 `virtual unsigned char ac-
tivemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller::getDataStructureT
() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.216.3.3 `virtual void ac-
tivemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller::looseMarshal
(OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut) throw (
decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 618).

6.216.3.4 virtual void activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 619).

6.216.3.5 virtual int activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 620).

6.216.3.6 virtual void activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 621).

```
6.216.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 622).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/ConnectionInfoMarshaller.h

6.217 activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1142).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ConnectionInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller**:

Public Member Functions

- **ConnectionInfoMarshaller** ()
- virtual **~ConnectionInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.217.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1142). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.217.2 Constructor & Destructor Documentation

6.217.2.1 `activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller::ConnectionInfoM
() [inline]`

6.217.2.2 `virtual
activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller::~~ConnectionInfo
() [inline, virtual]`

6.217.3 Member Function Documentation

6.217.3.1 `virtual commands::DataStructure* ac-
tivemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller::createObject
() const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.217.3.2 `virtual unsigned char ac-
tivemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller::getDataStructureT
() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.217.3.3 `virtual void ac-
tivemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller::looseMarshal
(OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut) throw (
decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 611).

6.217.3.4 virtual void activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 612).

6.217.3.5 virtual int activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 613).

6.217.3.6 virtual void activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 614).

```
6.217.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 615).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/ConnectionInfoMarshaller.h`

6.218 `activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller` Class Reference

Marshaling code for Open Wire Format for `ConnectionInfoMarshaller` (p. 1146).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ConnectionInfoMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller`:

Public Member Functions

- **ConnectionInfoMarshaller** ()
- virtual **~ConnectionInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.218.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1146). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.218.2 Constructor & Destructor Documentation

6.218.2.1 `activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller::ConnectionInfoM
() [inline]`

6.218.2.2 `virtual
activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller::~~ConnectionInfo
() [inline, virtual]`

6.218.3 Member Function Documentation

6.218.3.1 `virtual commands::DataStructure* ac-
tivemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller::createObject
() const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.218.3.2 `virtual unsigned char ac-
tivemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller::getDataStructureT
() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.218.3.3 `virtual void ac-
tivemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller::looseMarshal
(OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut) throw (
decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 624).

6.218.3.4 virtual void activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 625).

6.218.3.5 virtual int activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 626).

6.218.3.6 virtual void activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 628).

```
6.218.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 629).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ConnectionInfoMarshaller.h`

6.219 **activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller** Class Reference

Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1150).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ConnectionInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller**:

Public Member Functions

- **ConnectionInfoMarshaller** ()
- virtual **~ConnectionInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.219.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1150). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.219.2 Constructor & Destructor Documentation

6.219.2.1 `activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller::ConnectionInfoM`
`() [inline]`

6.219.2.2 `virtual`
`activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller::~~ConnectionInfo`
`() [inline, virtual]`

6.219.3 Member Function Documentation

6.219.3.1 `virtual commands::DataStructure* ac-`
`tivemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller::createObject`
`() const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.219.3.2 `virtual unsigned char ac-`
`tivemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller::getDataStructureT`
`() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.219.3.3 `virtual void ac-`
`tivemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller::looseMarshal`
`(OpenWireFormat * wireFormat, commands::DataStructure *`
`dataStructure, decaf::io::DataOutputStream * dataOut) throw (`
`decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 631).

6.219.3.4 virtual void activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 632).

6.219.3.5 virtual int activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 633).

6.219.3.6 virtual void activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 634).

```
6.219.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 635).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/ConnectionInfoMarshaller.h

6.220 cms::ConnectionMetaData Class Reference

A **ConnectionMetaData** (p. 1154) object provides information describing the **Connection** (p. 1052) object.

```
#include <src/main/cms/ConnectionMetaData.h>
```

Inheritance diagram for cms::ConnectionMetaData:

Public Member Functions

- virtual `~ConnectionMetaData ()`
- virtual `std::string getCMSVersion () const =0 throw (cms::CMSEException)`
Gets the CMS API version.
- virtual `int getCMSMajorVersion () const =0 throw (cms::CMSEException)`
Gets the CMS major version number.
- virtual `int getCMSMinorVersion () const =0 throw (cms::CMSEException)`
Gets the CMS minor version number.
- virtual `std::string getCMSProviderName () const =0 throw (cms::CMSEException)`
Gets the CMS provider name.
- virtual `std::string getProviderVersion () const =0 throw (cms::CMSEException)`
Gets the CMS provider version.
- virtual `int getProviderMajorVersion () const =0 throw (cms::CMSEException)`
Gets the CMS provider major version number.
- virtual `int getProviderMinorVersion () const =0 throw (cms::CMSEException)`
Gets the CMS provider minor version number.
- virtual `std::vector< std::string > getCMSXPropertyNames () const =0 throw (cms::CMSEException)`
Gets an Vector of the CMSX property names.

6.220.1 Detailed Description

A **ConnectionMetaData** (p.1154) object provides information describing the **Connection** (p.1052) object.

Since

1.3

6.220.2 Constructor & Destructor Documentation

- 6.220.2.1** `virtual cms::ConnectionMetaData::~~ConnectionMetaData () [inline, virtual]`

6.220.3 Member Function Documentation

- 6.220.3.1** `virtual int cms::ConnectionMetaData::getCMSMajorVersion () const throw (cms::CMSEException) [pure virtual]`

Gets the CMS major version number.

Returns

the CMS API major version number

Exceptions

CMSEException (p. 960) If the CMS Provider fails to retrieve the metadata due to some internal error.

Implemented in `activemq::core::ActiveMQConnectionMetaData` (p. 217).

6.220.3.2 `virtual int cms::ConnectionMetaData::getCMSMinorVersion () const throw (cms::CMSEException) [pure virtual]`

Gets the CMS minor version number.

Returns

the CMS API minor version number

Exceptions

CMSEException (p. 960) If the CMS Provider fails to retrieve the metadata due to some internal error.

Implemented in `activemq::core::ActiveMQConnectionMetaData` (p. 218).

6.220.3.3 `virtual std::string cms::ConnectionMetaData::getCMSProviderName () const throw (cms::CMSEException) [pure virtual]`

Gets the CMS provider name.

Returns

the CMS provider name

Exceptions

CMSEException (p. 960) If the CMS Provider fails to retrieve the metadata due to some internal error.

Implemented in `activemq::core::ActiveMQConnectionMetaData` (p. 218).

6.220.3.4 `virtual std::string cms::ConnectionMetaData::getCMSVersion () const throw (cms::CMSEException) [pure virtual]`

Gets the CMS API version.

Returns

the CMS API Version in String form.

Exceptions

CMSEException (p. 960) If the CMS Provider fails to retrieve the metadata due to some internal error.

Implemented in `activemq::core::ActiveMQConnectionMetaData` (p. 218).

6.220.3.5 `virtual std::vector<std::string>
 cms::ConnectionMetaData::getCMSXPropertyNames ()
 const throw (cms::CMSEException) [pure virtual]`

Gets an Vector of the CMSX property names.

Returns

an Vector of CMSX property names

Exceptions

CMSEException (p. 960) If the CMS Provider fails to retrieve the metadata due to some internal error.

Implemented in `activemq::core::ActiveMQConnectionMetaData` (p. 219).

6.220.3.6 `virtual int cms::ConnectionMetaData::getProviderMajorVersion ()
 const throw (cms::CMSEException) [pure virtual]`

Gets the CMS provider major version number.

Returns

the CMS provider major version number

Exceptions

CMSEException (p. 960) If the CMS Provider fails to retrieve the metadata due to some internal error.

Implemented in `activemq::core::ActiveMQConnectionMetaData` (p. 219).

6.220.3.7 `virtual int cms::ConnectionMetaData::getProviderMinorVersion ()
 const throw (cms::CMSEException) [pure virtual]`

Gets the CMS provider minor version number.

Returns

the CMS provider minor version number

Exceptions

CMSEException (p. 960) If the CMS Provider fails to retrieve the metadata due to some internal error.

Implemented in `activemq::core::ActiveMQConnectionMetaData` (p. 219).

6.220.3.8 `virtual std::string cms::ConnectionMetaData::getProviderVersion ()
 const throw (cms::CMSEException) [pure virtual]`

Gets the CMS provider version.

Returns

the CMS provider version

Exceptions

CMSException (p. 960) If the CMS Provider fails to retrieve the metadata due to some internal error.

Implemented in **activemq::core::ActiveMQConnectionMetaData** (p. 220).

The documentation for this class was generated from the following file:

- `src/main/cms/ConnectionMetaData.h`

6.221 **activemq::state::ConnectionState** Class Reference

```
#include <src/main/activemq/state/ConnectionState.h>
```

Public Member Functions

- **ConnectionState** (const **Pointer**< **ConnectionInfo** > &info)
- virtual **~ConnectionState** ()
- std::string **toString** () const
- const **Pointer**< **commands::ConnectionInfo** > & **getInfo** () const
- void **checkShutdown** () const
- void **shutdown** ()
- void **reset** (const **Pointer**< **ConnectionInfo** > &info)
- void **addTempDestination** (const **Pointer**< **DestinationInfo** > &info)
- void **removeTempDestination** (const **Pointer**< **ActiveMQDestination** > &destination)
- void **addTransactionState** (const **Pointer**< **TransactionId** > &id)
- const **Pointer**< **TransactionState** > & **getTransactionState** (const **Pointer**< **TransactionId** > &id) const
- std::vector< **Pointer**< **TransactionState** > > **getTransactionStates** () const
- **Pointer**< **TransactionState** > **removeTransactionState** (const **Pointer**< **TransactionId** > &id)
- void **addSession** (const **Pointer**< **SessionInfo** > &info)
- **Pointer**< **SessionState** > **removeSession** (const **Pointer**< **SessionId** > &id)
- const **Pointer**< **SessionState** > & **getSessionState** (const **Pointer**< **SessionId** > &id) const
- const **StlList**< **Pointer**< **DestinationInfo** > > & **getTempDesinations** () const
- std::vector< **Pointer**< **SessionState** > > **getSessionStates** () const

6.221.1 Constructor & Destructor Documentation

6.221.1.1 `activemq::state::ConnectionState::ConnectionState (const Pointer< ConnectionInfo > & info)`

6.221.1.2 `virtual activemq::state::ConnectionState::~~ConnectionState ()`
[virtual]

6.221.2 Member Function Documentation

6.221.2.1 `void activemq::state::ConnectionState::addSession (const Pointer< SessionInfo > & info)` [inline]

6.221.2.2 `void activemq::state::ConnectionState::addTempDestination (const Pointer< DestinationInfo > & info)` [inline]

6.221.2.3 `void activemq::state::ConnectionState::addTransactionState (const Pointer< TransactionId > & id)` [inline]

6.221.2.4 `void activemq::state::ConnectionState::checkShutdown ()` const

6.221.2.5 `const Pointer< commands::ConnectionInfo > & activemq::state::ConnectionState::getInfo ()` const [inline]

6.221.2.6 `const Pointer< SessionState > & activemq::state::ConnectionState::getSessionState (const Pointer< SessionId > & id)` const [inline]

6.221.2.7 `std::vector< Pointer< SessionState > > activemq::state::ConnectionState::getSessionStates ()` const [inline]

6.221.2.8 `const StlList< Pointer< DestinationInfo > > & activemq::state::ConnectionState::getTempDesinations ()` const [inline]

6.221.2.9 `const Pointer< TransactionState > & activemq::state::ConnectionState::getTransactionState (const Pointer< TransactionId > & id)` const [inline]

6.221.2.10 `std::vector< Pointer< TransactionState > > activemq::state::ConnectionState::getTransactionStates ()` const [inline]

6.221.2.11 `Pointer< SessionState > activemq::state::ConnectionState::removeSession (const Pointer< SessionId > & id)` [inline]

6.221.2.12 `void activemq::state::ConnectionState::removeTempDestination (const Pointer< ActiveMQDestination > & destination)` [inline]

References `decaf::lang::Pointer< T, REFCOUNTER >::get()`.

- 6.221.2.13** `Pointer<TransactionState> activemq::state::ConnectionState::removeTransactionState (const Pointer< TransactionId > & id) [inline]`
- 6.221.2.14** `void activemq::state::ConnectionState::reset (const Pointer< ConnectionInfo > & info)`
- 6.221.2.15** `void activemq::state::ConnectionState::shutdown ()`
- 6.221.2.16** `std::string activemq::state::ConnectionState::toString () const`

The documentation for this class was generated from the following file:

- `src/main/activemq/state/ConnectionState.h`

6.222 activemq::state::ConnectionStateTracker Class Reference

```
#include <src/main/activemq/state/ConnectionStateTracker.h>
```

Inheritance diagram for `activemq::state::ConnectionStateTracker`:

Public Member Functions

- `ConnectionStateTracker ()`
- `virtual ~ConnectionStateTracker ()`
- `Pointer< Tracked > track (const Pointer< Command > &command) throw (decaf::io::IOException)`
- `void trackBack (const Pointer< Command > &command)`
- `void restore (const Pointer< transport::Transport > &transport) throw (decaf::io::IOException)`
- `virtual Pointer< Command > processDestinationInfo (DestinationInfo *info) throw (exceptions::ActiveMQException)`
- `virtual Pointer< Command > processRemoveDestination (DestinationInfo *info) throw (exceptions::ActiveMQException)`
- `virtual Pointer< Command > processProducerInfo (ProducerInfo *info) throw (exceptions::ActiveMQException)`
- `virtual Pointer< Command > processRemoveProducer (ProducerId *id) throw (exceptions::ActiveMQException)`
- `virtual Pointer< Command > processConsumerInfo (ConsumerInfo *info) throw (exceptions::ActiveMQException)`
- `virtual Pointer< Command > processRemoveConsumer (ConsumerId *id) throw (exceptions::ActiveMQException)`
- `virtual Pointer< Command > processSessionInfo (SessionInfo *info) throw (exceptions::ActiveMQException)`
- `virtual Pointer< Command > processRemoveSession (SessionId *id) throw (exceptions::ActiveMQException)`
- `virtual Pointer< Command > processConnectionInfo (ConnectionInfo *info) throw (exceptions::ActiveMQException)`

- virtual **Pointer< Command > processRemoveConnection** (**ConnectionId** *id) throw (exceptions::ActiveMQException)
- virtual **Pointer< Command > processMessage** (**Message** *message) throw (exceptions::ActiveMQException)
- virtual **Pointer< Command > processMessageAck** (**MessageAck** *ack) throw (exceptions::ActiveMQException)
- virtual **Pointer< Command > processBeginTransaction** (**TransactionInfo** *info) throw (exceptions::ActiveMQException)
- virtual **Pointer< Command > processPrepareTransaction** (**TransactionInfo** *info) throw (exceptions::ActiveMQException)
- virtual **Pointer< Command > processCommitTransactionOnePhase** (**TransactionInfo** *info) throw (exceptions::ActiveMQException)
- virtual **Pointer< Command > processCommitTransactionTwoPhase** (**TransactionInfo** *info) throw (exceptions::ActiveMQException)
- virtual **Pointer< Command > processRollbackTransaction** (**TransactionInfo** *info) throw (exceptions::ActiveMQException)
- virtual **Pointer< Command > processEndTransaction** (**TransactionInfo** *info) throw (exceptions::ActiveMQException)
- bool **isRestoreConsumers** () const
- void **setRestoreConsumers** (bool restoreConsumers)
- bool **isRestoreProducers** () const
- void **setRestoreProducers** (bool restoreProducers)
- bool **isRestoreSessions** () const
- void **setRestoreSessions** (bool restoreSessions)
- bool **isTrackTransactions** () const
- void **setTrackTransactions** (bool trackTransactions)
- bool **isRestoreTransaction** () const
- void **setRestoreTransaction** (bool restoreTransaction)
- bool **isTrackMessages** () const
- void **setTrackMessages** (bool trackMessages)
- int **getMaxCacheSize** () const
- void **setMaxCacheSize** (int maxCacheSize)

Friends

- class **RemoveTransactionAction**

6.222.1 Constructor & Destructor Documentation

6.222.1.1 `activemq::state::ConnectionStateTracker::ConnectionStateTracker ()`

6.222.1.2 `virtual
activemq::state::ConnectionStateTracker::~~ConnectionStateTracker ()
[virtual]`

6.222.2 Member Function Documentation

6.222.2.1 `int activemq::state::ConnectionStateTracker::getMaxCacheSize ()
const [inline]`

6.222.2.2 `bool activemq::state::ConnectionStateTracker::isRestoreConsumers ()
const [inline]`

6.222.2.3 `bool activemq::state::ConnectionStateTracker::isRestoreProducers ()
const [inline]`

6.222.2.4 `bool activemq::state::ConnectionStateTracker::isRestoreSessions ()
const [inline]`

6.222.2.5 `bool activemq::state::ConnectionStateTracker::isRestoreTransaction ()
const [inline]`

6.222.2.6 `bool activemq::state::ConnectionStateTracker::isTrackMessages ()
const [inline]`

6.222.2.7 `bool activemq::state::ConnectionStateTracker::isTrackTransactions ()
const [inline]`

6.222.2.8 `virtual Pointer<Command> ac-
tivemq::state::ConnectionStateTracker::processBeginTransaction (
TransactionInfo * info) throw (exceptions::ActiveMQException)
[virtual]`

Implements `activemq::state::CommandVisitor` (p. 998).

6.222.2.9 `virtual Pointer<Command> ac-
tivemq::state::ConnectionStateTracker::processCommitTransactionOnePhase
(TransactionInfo * info) throw (exceptions::ActiveMQException)
[virtual]`

Implements `activemq::state::CommandVisitor` (p. 999).

6.222.2.10 `virtual Pointer<Command> ac-
tivemq::state::ConnectionStateTracker::processCommitTransactionTwoPhase
(TransactionInfo * info) throw (exceptions::ActiveMQException)
[virtual]`

Implements `activemq::state::CommandVisitor` (p. 999).

6.222.2.11 virtual Pointer<Command> activemq::state::ConnectionStateTracker::processConnectionInfo (ConnectionInfo * *info*) throw (exceptions::ActiveMQException)
[virtual]

Implements activemq::state::CommandVisitor (p. 999).

6.222.2.12 virtual Pointer<Command> activemq::state::ConnectionStateTracker::processConsumerInfo (ConsumerInfo * *info*) throw (exceptions::ActiveMQException)
[virtual]

Implements activemq::state::CommandVisitor (p. 999).

6.222.2.13 virtual Pointer<Command> activemq::state::ConnectionStateTracker::processDestinationInfo (DestinationInfo * *info*) throw (exceptions::ActiveMQException)
[virtual]

Implements activemq::state::CommandVisitor (p. 1000).

6.222.2.14 virtual Pointer<Command> activemq::state::ConnectionStateTracker::processEndTransaction (TransactionInfo * *info*) throw (exceptions::ActiveMQException)
[virtual]

Implements activemq::state::CommandVisitor (p. 1000).

6.222.2.15 virtual Pointer<Command> activemq::state::ConnectionStateTracker::processMessage (Message * *message*) throw (exceptions::ActiveMQException)
[virtual]

Implements activemq::state::CommandVisitor (p. 1000).

6.222.2.16 virtual Pointer<Command> activemq::state::ConnectionStateTracker::processMessageAck (MessageAck * *ack*) throw (exceptions::ActiveMQException)
[virtual]

Implements activemq::state::CommandVisitor (p. 1000).

6.222.2.17 virtual Pointer<Command> activemq::state::ConnectionStateTracker::processPrepareTransaction (TransactionInfo * *info*) throw (exceptions::ActiveMQException)
[virtual]

Implements activemq::state::CommandVisitor (p. 1001).

6.222.2.18 `virtual Pointer<Command> activemq::state::ConnectionStateTracker::processProducerInfo (ProducerInfo * info) throw (exceptions::ActiveMQException)` [virtual]

Implements `activemq::state::CommandVisitor` (p. 1001).

6.222.2.19 `virtual Pointer<Command> activemq::state::ConnectionStateTracker::processRemoveConnection (ConnectionId * id) throw (exceptions::ActiveMQException)` [virtual]

Implements `activemq::state::CommandVisitor` (p. 1001).

6.222.2.20 `virtual Pointer<Command> activemq::state::ConnectionStateTracker::processRemoveConsumer (ConsumerId * id) throw (exceptions::ActiveMQException)` [virtual]

Implements `activemq::state::CommandVisitor` (p. 1001).

6.222.2.21 `virtual Pointer<Command> activemq::state::ConnectionStateTracker::processRemoveDestination (DestinationInfo * info) throw (exceptions::ActiveMQException)` [virtual]

Implements `activemq::state::CommandVisitor` (p. 1002).

6.222.2.22 `virtual Pointer<Command> activemq::state::ConnectionStateTracker::processRemoveProducer (ProducerId * id) throw (exceptions::ActiveMQException)` [virtual]

Implements `activemq::state::CommandVisitor` (p. 1002).

6.222.2.23 `virtual Pointer<Command> activemq::state::ConnectionStateTracker::processRemoveSession (SessionId * id) throw (exceptions::ActiveMQException)` [virtual]

Implements `activemq::state::CommandVisitor` (p. 1002).

6.222.2.24 `virtual Pointer<Command> activemq::state::ConnectionStateTracker::processRollbackTransaction (TransactionInfo * info) throw (exceptions::ActiveMQException)` [virtual]

Implements `activemq::state::CommandVisitor` (p. 1002).

6.222.2.25 virtual Pointer<Command> activemq::state::ConnectionStateTracker::processSessionInfo (SessionInfo * *info*) throw (exceptions::ActiveMQException) [virtual]

Implements activemq::state::CommandVisitor (p.1003).

6.222.2.26 void activemq::state::ConnectionStateTracker::restore (const Pointer<transport::Transport > & *transport*) throw (decaf::io::IOException)

6.222.2.27 void activemq::state::ConnectionStateTracker::setMaxCacheSize (int *maxCacheSize*) [inline]

6.222.2.28 void activemq::state::ConnectionStateTracker::setRestoreConsumers (bool *restoreConsumers*) [inline]

6.222.2.29 void activemq::state::ConnectionStateTracker::setRestoreProducers (bool *restoreProducers*) [inline]

6.222.2.30 void activemq::state::ConnectionStateTracker::setRestoreSessions (bool *restoreSessions*) [inline]

6.222.2.31 void activemq::state::ConnectionStateTracker::setRestoreTransaction (bool *restoreTransaction*) [inline]

6.222.2.32 void activemq::state::ConnectionStateTracker::setTrackMessages (bool *trackMessages*) [inline]

6.222.2.33 void activemq::state::ConnectionStateTracker::setTrackTransactions (bool *trackTransactions*) [inline]

6.222.2.34 Pointer<Tracked> activemq::state::ConnectionStateTracker::track (const Pointer< Command > & *command*) throw (decaf::io::IOException)

6.222.2.35 void activemq::state::ConnectionStateTracker::trackBack (const Pointer< Command > & *command*)

6.222.3 Friends And Related Function Documentation

6.222.3.1 friend class RemoveTransactionAction [friend]

The documentation for this class was generated from the following file:

- src/main/activemq/state/ConnectionStateTracker.h

6.223 activemq::commands::ConsumerControl Class Reference

```
#include <src/main/activemq/commands/ConsumerControl.h>
```

Inheritance diagram for `activemq::commands::ConsumerControl`:

Public Member Functions

- **ConsumerControl** ()
- virtual **~ConsumerControl** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **ConsumerControl * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1372) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent.*
- virtual bool **isClose** () const
- virtual void **setClose** (bool close)
- virtual const **Pointer**< **ConsumerId** > & **getConsumerId** () const
- virtual **Pointer**< **ConsumerId** > & **getConsumerId** ()
- virtual void **setConsumerId** (const **Pointer**< **ConsumerId** > &consumerId)
- virtual int **getPrefetch** () const
- virtual void **setPrefetch** (int prefetch)
- virtual bool **isFlush** () const
- virtual void **setFlush** (bool flush)
- virtual bool **isStart** () const
- virtual void **setStart** (bool start)
- virtual bool **isStop** () const
- virtual void **setStop** (bool stop)
- virtual **Pointer**< **Command** > **visit** (**activemq::state::CommandVisitor** *visitor)
throw (exceptions::ActiveMQException)
Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_CONSUMERCONTROL** = 17

Protected Member Functions

- **ConsumerControl** (const **ConsumerControl** &)
- **ConsumerControl** & **operator=** (const **ConsumerControl** &)

Protected Attributes

- bool **close**
- **Pointer**< **ConsumerId** > **consumerId**
- int **prefetch**
- bool **flush**
- bool **start**
- bool **stop**

6.223.1 Constructor & Destructor Documentation

6.223.1.1 **activemq::commands::ConsumerControl::ConsumerControl** (const **ConsumerControl** &) [inline, protected]

6.223.1.2 **activemq::commands::ConsumerControl::ConsumerControl** ()

6.223.1.3 **virtual activemq::commands::ConsumerControl::~~ConsumerControl** () [virtual]

6.223.2 Member Function Documentation

6.223.2.1 **virtual ConsumerControl* activemq::commands::ConsumerControl::cloneDataStructure** () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1373).

6.223.2.2 **virtual void activemq::commands::ConsumerControl::copyDataStructure** (const **DataStructure** * *src*) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 598).

6.223.2.3 `virtual bool activemq::commands::ConsumerControl::equals (const
DataStructure * value) const` [virtual]

Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 598).

6.223.2.4 `virtual const Pointer<ConsumerId>& ac-
tivemq::commands::ConsumerControl::getConsumerId (
) const` [virtual]

6.223.2.5 `virtual Pointer<ConsumerId>& ac-
tivemq::commands::ConsumerControl::getConsumerId (
)` [virtual]

6.223.2.6 `virtual unsigned char ac-
tivemq::commands::ConsumerControl::getDataStructureType () const`
[virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataStructure** (p. 1372) type copy.

Implements **activemq::commands::DataStructure** (p. 1375).

- 6.223.2.7 `virtual int activemq::commands::ConsumerControl::getPrefetch ()
const [virtual]`
- 6.223.2.8 `virtual bool activemq::commands::ConsumerControl::isClose () const
[virtual]`
- 6.223.2.9 `virtual bool activemq::commands::ConsumerControl::isFlush () const
[virtual]`
- 6.223.2.10 `virtual bool activemq::commands::ConsumerControl::isStart () const
[virtual]`
- 6.223.2.11 `virtual bool activemq::commands::ConsumerControl::isStop () const
[virtual]`
- 6.223.2.12 `ConsumerControl& activemq::commands::ConsumerControl::operator=
(const ConsumerControl &) [inline, protected]`
- 6.223.2.13 `virtual void activemq::commands::ConsumerControl::setClose (bool
close) [virtual]`
- 6.223.2.14 `virtual void activemq::commands::ConsumerControl::setConsumerId (const Pointer< ConsumerId > & consumerId) [virtual]`
- 6.223.2.15 `virtual void activemq::commands::ConsumerControl::setFlush (bool
flush) [virtual]`
- 6.223.2.16 `virtual void activemq::commands::ConsumerControl::setPrefetch (int
prefetch) [virtual]`
- 6.223.2.17 `virtual void activemq::commands::ConsumerControl::setStart (bool
start) [virtual]`
- 6.223.2.18 `virtual void activemq::commands::ConsumerControl::setStop (bool
stop) [virtual]`
- 6.223.2.19 `virtual std::string activemq::commands::ConsumerControl::toString () const
[virtual]`

Returns a string containing the information for this **DataSet** (p.1372) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 602).

6.223.2.20 `virtual Pointer<Command> activemq::commands::ConsumerControl::visit (activemq::state::CommandVisitor * visitor) throw (exceptions::ActiveMQException)` [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

Returns

a **Response** (p. 2611) to the visitor being called or NULL if no response.

Implements `activemq::commands::Command` (p. 995).

6.223.3 Field Documentation

6.223.3.1 `bool activemq::commands::ConsumerControl::close` [protected]

6.223.3.2 `Pointer<ConsumerId> activemq::commands::ConsumerControl::consumerId` [protected]

6.223.3.3 `bool activemq::commands::ConsumerControl::flush` [protected]

6.223.3.4 `const unsigned char activemq::commands::ConsumerControl::ID_CONSUMERCONTROL = 17` [static]

6.223.3.5 `int activemq::commands::ConsumerControl::prefetch` [protected]

6.223.3.6 `bool activemq::commands::ConsumerControl::start` [protected]

6.223.3.7 `bool activemq::commands::ConsumerControl::stop` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ConsumerControl.h`

6.224 `activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller` Class Reference

Marshaling code for Open Wire Format for `ConsumerControlMarshaller` (p. 1170).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ConsumerControlMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller`:

Public Member Functions

- **ConsumerControlMarshaller** ()
- virtual **~ConsumerControlMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.224.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p.1170). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.224.2 Constructor & Destructor Documentation

6.224.2.1 `activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller::ConsumerControlMarshaller () [inline]`

6.224.2.2 `virtual activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller::~~ConsumerControlMarshaller () [inline, virtual]`

6.224.3 Member Function Documentation

6.224.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.224.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.224.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 604).

6.224.3.4 virtual void activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 605).

6.224.3.5 virtual int activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 606).

6.224.3.6 virtual void activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 608).

```
6.224.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 609).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ConsumerControlMarshaller.h`

6.225 `activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller` Class Reference

Marshaling code for Open Wire Format for `ConsumerControlMarshaller` (p. 1174).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ConsumerControlMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller`:

Public Member Functions

- **ConsumerControlMarshaller** ()
- virtual **~ConsumerControlMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.225.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p.1174). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.225.2 Constructor & Destructor Documentation

6.225.2.1 `activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller::ConsumerControlMarshaller () [inline]`

6.225.2.2 `virtual activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller::~~ConsumerControlMarshaller () [inline, virtual]`

6.225.3 Member Function Documentation

6.225.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.225.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.225.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 618).

6.225.3.4 virtual void activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 619).

6.225.3.5 virtual int activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 620).

6.225.3.6 virtual void activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 621).

```
6.225.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 622).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/ConsumerControlMarshaller.h`

6.226 `activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller` Class Reference

Marshaling code for Open Wire Format for `ConsumerControlMarshaller` (p. 1178).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ConsumerControlMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller`:

Public Member Functions

- **ConsumerControlMarshaller** ()
- virtual **~ConsumerControlMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.226.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p.1178). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.226.2 Constructor & Destructor Documentation

6.226.2.1 `activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller::ConsumerControlMarshaller () [inline]`

6.226.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller::~~ConsumerControlMarshaller () [inline, virtual]`

6.226.3 Member Function Documentation

6.226.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.226.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.226.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 611).

6.226.3.4 virtual void activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 612).

6.226.3.5 virtual int activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 613).

6.226.3.6 virtual void activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 614).

```
6.226.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 615).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/ConsumerControlMarshaller.h`

6.227 `activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller` Class Reference

Marshaling code for Open Wire Format for `ConsumerControlMarshaller` (p. 1182).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ConsumerControlMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller`:

Public Member Functions

- **ConsumerControlMarshaller** ()
- virtual **~ConsumerControlMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.227.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p.1182). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.227.2 Constructor & Destructor Documentation

6.227.2.1 `activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller::ConsumerControlMarshaller () [inline]`

6.227.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller::~~ConsumerControlMarshaller () [inline, virtual]`

6.227.3 Member Function Documentation

6.227.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.227.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.227.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 624).

6.227.3.4 virtual void activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 625).

6.227.3.5 virtual int activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 626).

6.227.3.6 virtual void activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 628).

```
6.227.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 629).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ConsumerControlMarshaller.h`

6.228 `activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller` Class Reference

Marshaling code for Open Wire Format for `ConsumerControlMarshaller` (p. 1186).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ConsumerControlMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller`:

Public Member Functions

- **ConsumerControlMarshaller** ()
- virtual **~ConsumerControlMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.228.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p.1186). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.228.2 Constructor & Destructor Documentation

6.228.2.1 `activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller::ConsumerControlMarshaller () [inline]`

6.228.2.2 `virtual activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller::~~ConsumerControlMarshaller () [inline, virtual]`

6.228.3 Member Function Documentation

6.228.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.228.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.228.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 631).

6.228.3.4 virtual void activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 632).

6.228.3.5 virtual int activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 633).

6.228.3.6 virtual void activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions

- IOException* if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 634).

```
6.228.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 635).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**ConsumerControlMarshaller.h**

6.229 activemq::commands::ConsumerId Class Reference

```
#include <src/main/activemq/commands/ConsumerId.h>
```

Inheritance diagram for **activemq::commands::ConsumerId**:

Public Types

- typedef **decaf::lang::PointerComparator**< **ConsumerId** > **COMPARATOR**

Public Member Functions

- **ConsumerId** ()
- **ConsumerId** (const **ConsumerId** &other)
- **ConsumerId** (const **SessionId** &sessionId, long long consumerId)
- virtual ~**ConsumerId** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **ConsumerId** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1372) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent.*
- const **Pointer**< **SessionId** > & **getParentId** () const
- virtual const std::string & **getConnectionId** () const
- virtual std::string & **getConnectionId** ()
- virtual void **setConnectionId** (const std::string &connectionId)
- virtual long long **getSessionId** () const
- virtual void **setSessionId** (long long sessionId)
- virtual long long **getValue** () const
- virtual void **setValue** (long long value)
- virtual int **compareTo** (const **ConsumerId** &value) const
- virtual bool **equals** (const **ConsumerId** &value) const
- virtual bool **operator==** (const **ConsumerId** &value) const
- virtual bool **operator<** (const **ConsumerId** &value) const
- **ConsumerId** & **operator=** (const **ConsumerId** &other)

Static Public Attributes

- static const unsigned char **ID_CONSUMERID** = 122

Protected Attributes

- std::string **connectionId**
- long long **sessionId**
- long long **value**

6.229.1 Member Typedef Documentation

6.229.1.1 `typedef decaf::lang::PointerComparator<ConsumerId>
activemq::commands::ConsumerId::COMPARATOR`

6.229.2 Constructor & Destructor Documentation

6.229.2.1 `activemq::commands::ConsumerId::ConsumerId ()`

6.229.2.2 `activemq::commands::ConsumerId::ConsumerId (const ConsumerId &
other)`

6.229.2.3 `activemq::commands::ConsumerId::ConsumerId (const SessionId &
sessionId, long long consumerId) [inline]`

References `activemq::commands::SessionId::getConnectionId()`, and `activemq::commands::SessionId::getValue()`.

6.229.2.4 `virtual activemq::commands::ConsumerId::~~ConsumerId () [virtual]`

6.229.3 Member Function Documentation

6.229.3.1 `virtual ConsumerId* activemq::commands::ConsumerId::cloneDataStructure (const) [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

6.229.3.2 `virtual int activemq::commands::ConsumerId::compareTo (const ConsumerId & value) const [virtual]`

6.229.3.3 `virtual void activemq::commands::ConsumerId::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

6.229.3.4 `virtual bool activemq::commands::ConsumerId::equals (const DataStructure * value) const [virtual]`

Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

6.229.3.5 virtual bool activemq::commands::ConsumerId::equals (const ConsumerId & *value*) const [virtual]

6.229.3.6 virtual std::string& activemq::commands::ConsumerId::getConnectionId () [virtual]

6.229.3.7 virtual const std::string& activemq::commands::ConsumerId::getConnectionId () const [virtual]

6.229.3.8 virtual unsigned char activemq::commands::ConsumerId::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataStructure** (p. 1372) type copy.

- 6.229.3.9 `const Pointer<SessionId>& activemq::commands::ConsumerId::getParentId ()`
`const`
- 6.229.3.10 `virtual long long activemq::commands::ConsumerId::getSessionId ()`
`const` [virtual]
- 6.229.3.11 `virtual long long activemq::commands::ConsumerId::getValue ()`
`const` [virtual]
- 6.229.3.12 `virtual bool activemq::commands::ConsumerId::operator< (const ConsumerId & value) const` [virtual]
- 6.229.3.13 `ConsumerId& activemq::commands::ConsumerId::operator= (const ConsumerId & other)`
- 6.229.3.14 `virtual bool activemq::commands::ConsumerId::operator== (const ConsumerId & value) const` [virtual]
- 6.229.3.15 `virtual void activemq::commands::ConsumerId::setConnectionId (const std::string & connectionId)` [virtual]
- 6.229.3.16 `virtual void activemq::commands::ConsumerId::setSessionId (long long sessionId)` [virtual]
- 6.229.3.17 `virtual void activemq::commands::ConsumerId::setValue (long long value)` [virtual]
- 6.229.3.18 `virtual std::string activemq::commands::ConsumerId::toString ()`
`const` [virtual]

Returns a string containing the information for this **DataStructure** (p.1372) such as its type and value of its elements.

Returns

formatted string useful for debugging.

6.229.4 Field Documentation

- 6.229.4.1 `std::string activemq::commands::ConsumerId::connectionId` [protected]
- 6.229.4.2 `const unsigned char activemq::commands::ConsumerId::ID_-CONSUMERID = 122` [static]

Referenced by `activemq::state::CommandVisitorAdapter::processRemoveInfo()`.

- 6.229.4.3 `long long activemq::commands::ConsumerId::sessionId` [protected]
- 6.229.4.4 `long long activemq::commands::ConsumerId::value` [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/ConsumerId.h

6.230 activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller Class Reference

Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p.1195).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ConsumerIdMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller:

Public Member Functions

- **ConsumerIdMarshaller** ()
- virtual **~ConsumerIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.230.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1195). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.230.2 Constructor & Destructor Documentation

6.230.2.1 `activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller::ConsumerIdMarshaller () [inline]`

6.230.2.2 `virtual activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller::~~ConsumerIdMarshaller () [inline, virtual]`

6.230.3 Member Function Documentation

6.230.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.230.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.230.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1342).

```
6.230.3.4 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1348).

```
6.230.3.5 virtual int ac-
tivemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, utils::BooleanStream * bs ) throw (
decaf::io::IOException ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1354).

6.230.3.6 `virtual void activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1360).

6.230.3.7 `virtual void activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1366).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ConsumerIdMarshaller.h`

6.231 `activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller` Class Reference

Marshaling code for Open Wire Format for `ConsumerIdMarshaller` (p. 1198).

#include <src/main/activemq/wireformat/openwire/marshal/v4/ConsumerIdMarshaller.h>

Inheritance diagram for activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller:

Public Member Functions

- **ConsumerIdMarshaller** ()
- virtual **~ConsumerIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.231.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p.1198). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.231.2 Constructor & Destructor Documentation

6.231.2.1 `activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller::ConsumerIdMarshaller () [inline]`

6.231.2.2 `virtual activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller::~~ConsumerIdMarshaller () [inline, virtual]`

6.231.3 Member Function Documentation

6.231.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.231.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.231.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1342).

6.231.3.4 virtual void activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1348).

6.231.3.5 virtual int activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1354).

6.231.3.6 virtual void activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1360).

```
6.231.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1366).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/ConsumerIdMarshaller.h`

6.232 **activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller** Class Reference

Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1202).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ConsumerIdMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller**:

Public Member Functions

- **ConsumerIdMarshaller** ()
- virtual **~ConsumerIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.232.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p.1202). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.232.2 Constructor & Destructor Documentation

6.232.2.1 `activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller::ConsumerIdMarshaller () [inline]`

6.232.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller::~~ConsumerIdMarshaller () [inline, virtual]`

6.232.3 Member Function Documentation

6.232.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.232.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.232.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1342).

6.232.3.4 virtual void activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1348).

6.232.3.5 virtual int activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1354).

6.232.3.6 virtual void activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1360).

```
6.232.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1366).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/ConsumerIdMarshaller.h

6.233 activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller

Class Reference

Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1206).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ConsumerIdMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller**:

Public Member Functions

- **ConsumerIdMarshaller** ()
- virtual **~ConsumerIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.233.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p.1206). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.233.2 Constructor & Destructor Documentation

6.233.2.1 `activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller::ConsumerIdMarshaller () [inline]`

6.233.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller::~~ConsumerIdMarshaller () [inline, virtual]`

6.233.3 Member Function Documentation

6.233.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.233.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.233.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1342).

6.233.3.4 virtual void activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1348).

6.233.3.5 virtual int activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1354).

6.233.3.6 virtual void activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1360).

```
6.233.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1366).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ConsumerIdMarshaller.h`

6.234 **activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller** Class Reference

Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1210).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ConsumerIdMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller**:

Public Member Functions

- **ConsumerIdMarshaller** ()
- virtual **~ConsumerIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.234.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p.1210). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.234.2 Constructor & Destructor Documentation

6.234.2.1 `activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller::ConsumerIdMarshaller () [inline]`

6.234.2.2 `virtual activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller::~~ConsumerIdMarshaller () [inline, virtual]`

6.234.3 Member Function Documentation

6.234.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.234.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.234.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1342).

6.234.3.4 virtual void activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1348).

6.234.3.5 virtual int activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1354).

6.234.3.6 virtual void activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1360).

```
6.234.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1366).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**ConsumerIdMarshaller.h**

6.235 activemq::commands::ConsumerInfo Class Reference

```
#include <src/main/activemq/commands/ConsumerInfo.h>
```

Inheritance diagram for **activemq::commands::ConsumerInfo**:

Public Member Functions

- **ConsumerInfo** ()
- virtual **~ConsumerInfo** ()
- virtual unsigned char **getDataStructureType** () const

Get the unique identifier that this object and its own Marshaler share.

- virtual **ConsumerInfo** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1372) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **ConsumerId** > & **getConsumerId** () const
- virtual **Pointer**< **ConsumerId** > & **getConsumerId** ()
- virtual void **setConsumerId** (const **Pointer**< **ConsumerId** > &consumerId)
- virtual bool **isBrowser** () const
- virtual void **setBrowser** (bool browser)
- virtual const **Pointer**< **ActiveMQDestination** > & **getDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &destination)
- virtual int **getPrefetchSize** () const
- virtual void **setPrefetchSize** (int prefetchSize)
- virtual int **getMaximumPendingMessageLimit** () const
- virtual void **setMaximumPendingMessageLimit** (int maximumPendingMessageLimit)
- virtual bool **isDispatchAsync** () const
- virtual void **setDispatchAsync** (bool dispatchAsync)
- virtual const std::string & **getSelector** () const
- virtual std::string & **getSelector** ()
- virtual void **setSelector** (const std::string &selector)
- virtual const std::string & **getSubscriptionName** () const
- virtual std::string & **getSubscriptionName** ()
- virtual void **setSubscriptionName** (const std::string &subscriptionName)
- virtual bool **isNoLocal** () const
- virtual void **setNoLocal** (bool noLocal)
- virtual bool **isExclusive** () const
- virtual void **setExclusive** (bool exclusive)
- virtual bool **isRetroactive** () const
- virtual void **setRetroactive** (bool retroactive)
- virtual unsigned char **getPriority** () const
- virtual void **setPriority** (unsigned char priority)
- virtual const std::vector< **decaf::lang::Pointer**< **BrokerId** > > & **getBrokerPath** () const
- virtual std::vector< **decaf::lang::Pointer**< **BrokerId** > > & **getBrokerPath** ()
- virtual void **setBrokerPath** (const std::vector< **decaf::lang::Pointer**< **BrokerId** > > &brokerPath)
- virtual const **Pointer**< **BooleanExpression** > & **getAdditionalPredicate** () const

- virtual **Pointer**< **BooleanExpression** > & **getAdditionalPredicate** ()
- virtual void **setAdditionalPredicate** (const **Pointer**< **BooleanExpression** > &**additionalPredicate**)
- virtual bool **isNetworkSubscription** () const
- virtual void **setNetworkSubscription** (bool **networkSubscription**)
- virtual bool **isOptimizedAcknowledge** () const
- virtual void **setOptimizedAcknowledge** (bool **optimizedAcknowledge**)
- virtual bool **isNoRangeAcks** () const
- virtual void **setNoRangeAcks** (bool **noRangeAcks**)
- virtual const std::vector< **decaf::lang::Pointer**< **ConsumerId** > > & **getNetworkConsumerPath** () const
- virtual std::vector< **decaf::lang::Pointer**< **ConsumerId** > > & **getNetworkConsumerPath** ()
- virtual void **setNetworkConsumerPath** (const std::vector< **decaf::lang::Pointer**< **ConsumerId** > > &**networkConsumerPath**)
- virtual bool **isConsumerInfo** () const
- virtual **Pointer**< **Command** > **visit** (**activemq::state::CommandVisitor** ***visitor**)
throw (**exceptions::ActiveMQException**)

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_CONSUMERINFO** = 5

Protected Member Functions

- **ConsumerInfo** (const **ConsumerInfo** &)
- **ConsumerInfo** & **operator=** (const **ConsumerInfo** &)

Protected Attributes

- **Pointer**< **ConsumerId** > **consumerId**
- bool **browser**
- **Pointer**< **ActiveMQDestination** > **destination**
- int **prefetchSize**
- int **maximumPendingMessageLimit**
- bool **dispatchAsync**
- std::string **selector**
- std::string **subscriptionName**
- bool **noLocal**
- bool **exclusive**
- bool **retroactive**
- unsigned char **priority**
- std::vector< **decaf::lang::Pointer**< **BrokerId** > > **brokerPath**
- **Pointer**< **BooleanExpression** > **additionalPredicate**
- bool **networkSubscription**
- bool **optimizedAcknowledge**
- bool **noRangeAcks**
- std::vector< **decaf::lang::Pointer**< **ConsumerId** > > **networkConsumerPath**

6.235.1 Constructor & Destructor Documentation

6.235.1.1 `activemq::commands::ConsumerInfo::ConsumerInfo (const ConsumerInfo &) [inline, protected]`

6.235.1.2 `activemq::commands::ConsumerInfo::ConsumerInfo ()`

6.235.1.3 `virtual activemq::commands::ConsumerInfo::~~ConsumerInfo () [virtual]`

6.235.2 Member Function Documentation

6.235.2.1 `virtual ConsumerInfo* activemq::commands::ConsumerInfo::cloneDataStructure () const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1373).

6.235.2.2 `virtual void activemq::commands::ConsumerInfo::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

Reimplemented from `activemq::commands::BaseCommand` (p. 598).

6.235.2.3 `virtual bool activemq::commands::ConsumerInfo::equals (const DataStructure * value) const [virtual]`

Compares the `DataStructure` (p. 1372) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 598).

- 6.235.2.4 `virtual const Pointer<BooleanExpression>& activemq::commands::ConsumerInfo::getAdditionalPredicate () const [virtual]`

- 6.235.2.5 `virtual Pointer<BooleanExpression>& activemq::commands::ConsumerInfo::getAdditionalPredicate () [virtual]`

- 6.235.2.6 `virtual const std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::ConsumerInfo::getBrokerPath () const [virtual]`

- 6.235.2.7 `virtual std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::ConsumerInfo::getBrokerPath () [virtual]`

- 6.235.2.8 `virtual Pointer<ConsumerId>& activemq::commands::ConsumerInfo::getConsumerId () [virtual]`

- 6.235.2.9 `virtual const Pointer<ConsumerId>& activemq::commands::ConsumerInfo::getConsumerId () const [virtual]`

- 6.235.2.10 `virtual unsigned char activemq::commands::ConsumerInfo::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataStructure** (p. 1372) type copy.

Implements **activemq::commands::DataStructure** (p. 1375).

- 6.235.2.11 `virtual const Pointer<ActiveMQDestination>& activemq::commands::ConsumerInfo::getDestination () const [virtual]`
- 6.235.2.12 `virtual Pointer<ActiveMQDestination>& activemq::commands::ConsumerInfo::getDestination () [virtual]`
- 6.235.2.13 `virtual int activemq::commands::ConsumerInfo::getMaximumPendingMessageLimit () const [virtual]`
- 6.235.2.14 `virtual const std::vector< decaf::lang::Pointer<ConsumerId> >& activemq::commands::ConsumerInfo::getNetworkConsumerPath () const [virtual]`
- 6.235.2.15 `virtual std::vector< decaf::lang::Pointer<ConsumerId> >& activemq::commands::ConsumerInfo::getNetworkConsumerPath () [virtual]`
- 6.235.2.16 `virtual int activemq::commands::ConsumerInfo::getPrefetchSize () const [virtual]`
- 6.235.2.17 `virtual unsigned char activemq::commands::ConsumerInfo::getPriority () const [virtual]`
- 6.235.2.18 `virtual const std::string& activemq::commands::ConsumerInfo::getSelector () const [virtual]`
- 6.235.2.19 `virtual std::string& activemq::commands::ConsumerInfo::getSelector () [virtual]`
- 6.235.2.20 `virtual const std::string& activemq::commands::ConsumerInfo::getSubscriptionName () const [virtual]`
- 6.235.2.21 `virtual std::string& activemq::commands::ConsumerInfo::getSubscriptionName () [virtual]`
- 6.235.2.22 `virtual bool activemq::commands::ConsumerInfo::isBrowser () const [virtual]`
- 6.235.2.23 `virtual bool activemq::commands::ConsumerInfo::isConsumerInfo () const [inline, virtual]`

Returns

an answer of true to the `isConsumerInfo()` (p. 1219) query.

Reimplemented from `activemq::commands::BaseCommand` (p. 600).

- 6.235.2.24 virtual bool activemq::commands::ConsumerInfo::isDispatchAsync ()
const [virtual]
- 6.235.2.25 virtual bool activemq::commands::ConsumerInfo::isExclusive () const
[virtual]
- 6.235.2.26 virtual bool activemq::commands::ConsumerInfo::isNetworkSubscription
() const [virtual]
- 6.235.2.27 virtual bool activemq::commands::ConsumerInfo::isNoLocal () const
[virtual]
- 6.235.2.28 virtual bool activemq::commands::ConsumerInfo::isNoRangeAcks ()
const [virtual]
- 6.235.2.29 virtual bool ac-
tivismq::commands::ConsumerInfo::isOptimizedAcknowledge () const
[virtual]
- 6.235.2.30 virtual bool activemq::commands::ConsumerInfo::isRetroactive ()
const [virtual]
- 6.235.2.31 ConsumerInfo& activemq::commands::ConsumerInfo::operator= (
const ConsumerInfo &) [inline, protected]
- 6.235.2.32 virtual void activemq::commands::ConsumerInfo::setAdditionalPredicate
(const Pointer< BooleanExpression > & *additionalPredicate*)
[virtual]
- 6.235.2.33 virtual void activemq::commands::ConsumerInfo::setBrokerPath (
const std::vector< decaf::lang::Pointer< BrokerId > > & *brokerPath*)
[virtual]
- 6.235.2.34 virtual void activemq::commands::ConsumerInfo::setBrowser (bool
browser) [virtual]
- 6.235.2.35 virtual void activemq::commands::ConsumerInfo::setConsumerId (
const Pointer< ConsumerId > & *consumerId*) [virtual]
- 6.235.2.36 virtual void activemq::commands::ConsumerInfo::setDestination (
const Pointer< ActiveMQDestination > & *destination*) [virtual]
- 6.235.2.37 virtual void activemq::commands::ConsumerInfo::setDispatchAsync (
bool *dispatchAsync*) [virtual]
- 6.235.2.38 virtual void activemq::commands::ConsumerInfo::setExclusive (bool
exclusive) [virtual]
- 6.235.2.39 virtual void ac-
tivismq::commands::ConsumerInfo::setMaximumPendingMessageLimit (
int *maximumPendingMessageLimit*) [virtual]
- 6.235.2.40 virtual void ac-
tivismq::commands::ConsumerInfo::setNetworkConsumerPath
(const std::vector< decaf::lang::Pointer< ConsumerId > > &
networkConsumerPath) [virtual]
- 6.235.2.41 virtual void ac-
tivismq::commands::ConsumerInfo::setNetworkSubscription (bool
networkSubscription) [virtual]
- 6.235.2.42 virtual void activemq::commands::ConsumerInfo::setNoLocal (bool

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 602).

```
6.235.2.51 virtual Pointer<Command> activemq::commands::ConsumerInfo::visit  
    ( activemq::state::CommandVisitor * visitor ) throw (   
    exceptions::ActiveMQException ) [virtual]
```

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

Returns

a **Response** (p. 2611) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 995).

6.235.3 Field Documentation

- 6.235.3.1 `Pointer<BooleanExpression> activemq::commands::ConsumerInfo::additionalPredicate`
[protected]
- 6.235.3.2 `std::vector< decaf::lang::Pointer<BrokerId> > activemq::commands::ConsumerInfo::brokerPath` [protected]
- 6.235.3.3 `bool activemq::commands::ConsumerInfo::browser` [protected]
- 6.235.3.4 `Pointer<ConsumerId> activemq::commands::ConsumerInfo::consumerId`
[protected]
- 6.235.3.5 `Pointer<ActiveMQDestination> activemq::commands::ConsumerInfo::destination`
[protected]
- 6.235.3.6 `bool activemq::commands::ConsumerInfo::dispatchAsync` [protected]
- 6.235.3.7 `bool activemq::commands::ConsumerInfo::exclusive` [protected]
- 6.235.3.8 `const unsigned char activemq::commands::ConsumerInfo::ID_ - CONSUMERINFO = 5` [static]
- 6.235.3.9 `int activemq::commands::ConsumerInfo::maximumPendingMessageLimit`
[protected]
- 6.235.3.10 `std::vector< decaf::lang::Pointer<ConsumerId> > activemq::commands::ConsumerInfo::networkConsumerPath`
[protected]
- 6.235.3.11 `bool activemq::commands::ConsumerInfo::networkSubscription`
[protected]
- 6.235.3.12 `bool activemq::commands::ConsumerInfo::noLocal` [protected]
- 6.235.3.13 `bool activemq::commands::ConsumerInfo::noRangeAcks` [protected]
- 6.235.3.14 `bool activemq::commands::ConsumerInfo::optimizedAcknowledge`
[protected]
- 6.235.3.15 `int activemq::commands::ConsumerInfo::prefetchSize` [protected]
- 6.235.3.16 `unsigned char activemq::commands::ConsumerInfo::priority`
[protected]
- 6.235.3.17 `bool activemq::commands::ConsumerInfo::retroactive` [protected]
- 6.235.3.18 `std::string activemq::commands::ConsumerInfo::selector` [protected]
- 6.235.3.19 `std::string activemq::commands::ConsumerInfo::subscriptionName`
[protected]

The documentation for this class was generated from the following file:

Generated on Mon Jan 24 2011 16:15:22 for activemq-cpp-3.1.0 by Doxygen

- src/main/activemq/commands/ConsumerInfo.h

6.236 activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1223).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ConsumerInfoMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller:

Public Member Functions

- **ConsumerInfoMarshaller** ()
- virtual **~ConsumerInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.236.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1223). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.236.2 Constructor & Destructor Documentation

6.236.2.1 `activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller::ConsumerInfoMarshaller () [inline]`

6.236.2.2 `virtual activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller::~~ConsumerInfoMarshaller () [inline, virtual]`

6.236.3 Member Function Documentation

6.236.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.236.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.236.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 604).

6.236.3.4 virtual void `activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller::looseUnmarshal` (`OpenWireFormat * wireFormat`, `commands::DataStructure * dataStructure`, `decaf::io::DataInputStream * dataIn`) throw (`decaf::io::IOException`) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 605).

6.236.3.5 virtual int `activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller::tightMarshal1` (`OpenWireFormat * wireFormat`, `commands::DataStructure * dataStructure`, `utils::BooleanStream * bs`) throw (`decaf::io::IOException`) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 606).

6.236.3.6 `virtual void activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 608).

6.236.3.7 `virtual void activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 609).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ConsumerInfoMarshaller.h`

6.237 `activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller` Class Reference

Marshaling code for Open Wire Format for `ConsumerInfoMarshaller` (p. 1226).

#include <src/main/activemq/wireformat/openwire/marshal/v1/ConsumerInfoMarshaller.h>

Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller:

Public Member Functions

- **ConsumerInfoMarshaller** ()
- virtual **~ConsumerInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.237.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p.1226). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.237.2 Constructor & Destructor Documentation

6.237.2.1 `activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller::ConsumerInfoMarshaller () [inline]`

6.237.2.2 `virtual
activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller::~~ConsumerInfoMarshaller () [inline, virtual]`

6.237.3 Member Function Documentation

6.237.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.237.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.237.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 611).

6.237.3.4 virtual void activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 612).

6.237.3.5 virtual int activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 613).

6.237.3.6 virtual void activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 614).

```
6.237.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 615).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/ConsumerInfoMarshaller.h`

6.238 `activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller` Class Reference

Marshaling code for Open Wire Format for `ConsumerInfoMarshaller` (p. 1230).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ConsumerInfoMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller`:

Public Member Functions

- **ConsumerInfoMarshaller** ()
- virtual **~ConsumerInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.238.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p.1230). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.238.2 Constructor & Destructor Documentation

6.238.2.1 `activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller::ConsumerInfoMar`
`() [inline]`

6.238.2.2 `virtual`
`activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller::~~ConsumerInfoMar`
`() [inline, virtual]`

6.238.3 Member Function Documentation

6.238.3.1 `virtual commands::DataStructure* ac-`
`tivemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller::createObject`
`() const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.238.3.2 `virtual unsigned char ac-`
`tivemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller::getDataStructureTy`
`() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.238.3.3 `virtual void ac-`
`tivemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller::looseMarshal`
`(OpenWireFormat * wireFormat, commands::DataStructure *`
`dataStructure, decaf::io::DataOutputStream * dataOut) throw (`
`decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 618).

6.238.3.4 virtual void activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 619).

6.238.3.5 virtual int activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 620).

6.238.3.6 virtual void activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 621).

```
6.238.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 622).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/ConsumerInfoMarshaller.h`

6.239 **activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller** Class Reference

Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1234).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ConsumerInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller**:

Public Member Functions

- **ConsumerInfoMarshaller** ()
- virtual **~ConsumerInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.239.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p.1234). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.239.2 Constructor & Destructor Documentation

6.239.2.1 `activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller::ConsumerInfoMarshaller()` [inline]

6.239.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller::~~ConsumerInfoMarshaller()` [inline, virtual]

6.239.3 Member Function Documentation

6.239.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.239.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.239.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 624).

6.239.3.4 virtual void activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 625).

6.239.3.5 virtual int activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 626).

6.239.3.6 virtual void activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 628).

```
6.239.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 629).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ConsumerInfoMarshaller.h`

6.240 `activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller` Class Reference

Marshaling code for Open Wire Format for `ConsumerInfoMarshaller` (p. 1238).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ConsumerInfoMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller`:

Public Member Functions

- **ConsumerInfoMarshaller** ()
- virtual **~ConsumerInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.240.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p.1238). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.240.2 Constructor & Destructor Documentation

6.240.2.1 `activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller::ConsumerInfoMarshaller () [inline]`

6.240.2.2 `virtual activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller::~~ConsumerInfoMarshaller () [inline, virtual]`

6.240.3 Member Function Documentation

6.240.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.240.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.240.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 631).

6.240.3.4 virtual void activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 632).

6.240.3.5 virtual int activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 633).

6.240.3.6 virtual void activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions

- IOException* if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 634).

```
6.240.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 635).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ConsumerInfoMarshaller.h`

6.241 activemq::state::ConsumerState Class Reference

```
#include <src/main/activemq/state/ConsumerState.h>
```

Public Member Functions

- **ConsumerState** (const **Pointer**< **ConsumerInfo** > &info)
- virtual **~ConsumerState** ()
- **std::string toString** () const
- const **Pointer**< **ConsumerInfo** > & **getInfo** () const

6.241.1 Constructor & Destructor Documentation

6.241.1.1 `activemq::state::ConsumerState::ConsumerState (const Pointer< ConsumerInfo > & info)`

6.241.1.2 `virtual activemq::state::ConsumerState::~~ConsumerState () [virtual]`

6.241.2 Member Function Documentation

6.241.2.1 `const Pointer<ConsumerInfo>& activemq::state::ConsumerState::getInfo () const [inline]`

6.241.2.2 `std::string activemq::state::ConsumerState::toString () const`

The documentation for this class was generated from the following file:

- `src/main/activemq/state/ConsumerState.h`

6.242 activemq::commands::ControlCommand Class Reference

```
#include <src/main/activemq/commands/ControlCommand.h>
```

Inheritance diagram for `activemq::commands::ControlCommand`:

Public Member Functions

- `ControlCommand ()`
- `virtual ~ControlCommand ()`
- `virtual unsigned char getDataStructureType () const`
Get the unique identifier that this object and its own Marshaler share.
- `virtual ControlCommand * cloneDataStructure () const`
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- `virtual void copyDataStructure (const DataStructure *src)`
Copy the contents of the passed object into this object's members, overwriting any existing data.
- `virtual std::string toString () const`
Returns a string containing the information for this DataStructure (p. 1372) such as its type and value of its elements.
- `virtual bool equals (const DataStructure *value) const`
Compares the DataStructure (p. 1372) passed in to this one, and returns if they are equivalent.
- `virtual const std::string & getCommand () const`
- `virtual std::string & getCommand ()`

- virtual void **setCommand** (const std::string &command)
- virtual **Pointer< Command > visit** (activemq::state::CommandVisitor *visitor) throw (exceptions::ActiveMQException)

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_CONTROLCOMMAND** = 14

Protected Member Functions

- **ControlCommand** (const **ControlCommand** &)
- **ControlCommand & operator=** (const **ControlCommand** &)

Protected Attributes

- std::string **command**

6.242.1 Constructor & Destructor Documentation

6.242.1.1 **activemq::commands::ControlCommand::ControlCommand** (const **ControlCommand** &) [inline, protected]

6.242.1.2 **activemq::commands::ControlCommand::ControlCommand** ()

6.242.1.3 **virtual activemq::commands::ControlCommand::~~ControlCommand** () [virtual]

6.242.2 Member Function Documentation

6.242.2.1 **virtual ControlCommand* activemq::commands::ControlCommand::cloneDataStructure** () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1373).

6.242.2.2 **virtual void activemq::commands::ControlCommand::copyDataStructure** (const **DataStructure** * *src*) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 598).

6.242.2.3 `virtual bool activemq::commands::ControlCommand::equals (const DataStructure * value) const` [virtual]

Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 598).

6.242.2.4 `virtual std::string& activemq::commands::ControlCommand::getCommand ()` [virtual]

6.242.2.5 `virtual const std::string& activemq::commands::ControlCommand::getCommand () const` [virtual]

6.242.2.6 `virtual unsigned char activemq::commands::ControlCommand::getDataStructureType () const` [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataStructure** (p. 1372) type copy.

Implements **activemq::commands::DataStructure** (p. 1375).

6.242.2.7 `ControlCommand& activemq::commands::ControlCommand::operator= (const ControlCommand &)` [inline, protected]

6.242.2.8 `virtual void activemq::commands::ControlCommand::setCommand (const std::string & command)` [virtual]

6.242.2.9 `virtual std::string activemq::commands::ControlCommand::toString () const` [virtual]

Returns a string containing the information for this **DataStructure** (p. 1372) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from `activemq::commands::BaseCommand` (p. 602).

6.242.2.10 `virtual Pointer<Command> activemq::commands::ControlCommand::visit (activemq::state::CommandVisitor * visitor) throw (exceptions::ActiveMQException) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

Returns

a **Response** (p. 2611) to the visitor being called or NULL if no response.

Implements `activemq::commands::Command` (p. 995).

6.242.3 Field Documentation

6.242.3.1 `std::string activemq::commands::ControlCommand::command [protected]`

6.242.3.2 `const unsigned char activemq::commands::ControlCommand::ID _ - CONTROLCOMMAND = 14 [static]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ControlCommand.h`

6.243 `activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller` Class Reference

Marshaling code for Open Wire Format for `ControlCommandMarshaller` (p. 1246).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ControlCommandMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller`:

Public Member Functions

- `ControlCommandMarshaller ()`
- `virtual ~ControlCommandMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaller.

- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.243.1 Detailed Description

Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p.1246). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.243.2 Constructor & Destructor Documentation

6.243.2.1 `activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller::ControlComm`
() [inline]

6.243.2.2 `virtual`
`activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller::~~ControlComm`
() [inline, virtual]

6.243.3 Member Function Documentation

6.243.3.1 `virtual commands::DataStructure* ac-`
`tivemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller::createObject`
() const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1331).

6.243.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1336).

6.243.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 604).

6.243.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 605).

6.243.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 606).

6.243.3.6 `virtual void activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 608).

```

6.243.3.7 virtual void activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]

```

Un-marshall an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 609).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ControlCommandMarshaller.h`

6.244 activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1250).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ControlCommandMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller**:

Public Member Functions

- **ControlCommandMarshaller** ()
- virtual **~ControlCommandMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.244.1 Detailed Description

Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p.1250). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.244.2 Constructor & Destructor Documentation

6.244.2.1 activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller::ControlComm
() [inline]

6.244.2.2 virtual
activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller::~~ControlComm
() [inline, virtual]

6.244.3 Member Function Documentation

6.244.3.1 virtual commands::DataStructure* ac-
tivismq::wireformat::openwire::marshal::v4::ControlCommandMarshaller::createObject
() const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1331).

6.244.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller::getDataStructureType() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.244.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 618).

6.244.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 619).

6.244.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 620).

6.244.3.6 `virtual void activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 621).

6.244.3.7 `virtual void activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 622).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/ControlCommandMarshaller.h`

6.245 `activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller` Class Reference

Marshaling code for Open Wire Format for `ControlCommandMarshaller` (p. 1254).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ControlCommandMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller`:

Public Member Functions

- `ControlCommandMarshaller ()`
- `virtual ~ControlCommandMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaler.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.245.1 Detailed Description

Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1254). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.245.2 Constructor & Destructor Documentation

6.245.2.1 **activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller::ControlComm**
() [inline]

6.245.2.2 **virtual**
activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller::~~ControlComm
() [inline, virtual]

6.245.3 Member Function Documentation

6.245.3.1 **virtual commands::DataStructure* ac-**
tivemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller::createObject
() const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1331).

6.245.3.2 **virtual unsigned char ac-**
tivemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller::getDataStructur
() const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.245.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 611).

6.245.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 612).

6.245.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 613).

6.245.3.6 virtual void `activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller::tightMarshal2`
(`OpenWireFormat` * *wireFormat*, `commands::DataStructure`
* *dataStructure*, `decaf::io::DataOutputStream` * *dataOut*,
`utils::BooleanStream` * *bs*) throw (`decaf::io::IOException`) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 614).

6.245.3.7 virtual void `activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller::tightUnmarshal`
(`OpenWireFormat` * *wireFormat*, `commands::DataStructure`
* *dataStructure*, `decaf::io::DataInputStream` * *dataIn*,
`utils::BooleanStream` * *bs*) throw (`decaf::io::IOException`) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 615).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/ControlCommandMarshaller.h`

6.246 **activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller** Class Reference

Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1258).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ControlCommandMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller**:

Public Member Functions

- **ControlCommandMarshaller** ()
- virtual **~ControlCommandMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

- virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.246.1 Detailed Description

Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p.1258). NOTE! This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.246.2 Constructor & Destructor Documentation

6.246.2.1 activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller::ControlCommandMarshaller () [inline]

6.246.2.2 virtual activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller::~~ControlCommandMarshaller () [inline, virtual]

6.246.3 Member Function Documentation

6.246.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1331).

6.246.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1336).

6.246.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 624).

6.246.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 625).

6.246.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 626).

```
6.246.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 628).

```
6.246.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 629).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ControlCommandMarshaller.h`

6.247 activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1262).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ControlCommandMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller**:

Public Member Functions

- **ControlCommandMarshaller** ()
- virtual **~ControlCommandMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.247.1 Detailed Description

Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p.1262). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.247.2 Constructor & Destructor Documentation

6.247.2.1 activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller::ControlCommandMarshaller () [inline]

6.247.2.2 virtual
activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller::~~ControlCommandMarshaller () [inline, virtual]

6.247.3 Member Function Documentation

6.247.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1331).

6.247.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1336).

6.247.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 631).

6.247.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 632).

6.247.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 633).

```
6.247.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 634).

```
6.247.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 635).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ControlCommandMarshaller.h`

6.248 decaf::util::concurrent::CountDownLatch Class Reference

```
#include <src/main/decaf/util/concurrent/CountDownLatch.h>
```

Public Member Functions

- **CountDownLatch** (int count)
Constructor.
- virtual **~CountDownLatch** ()
- virtual void **await** () throw (lang::Exception)
Waits for the Count to be zero, and then returns.
- virtual bool **await** (unsigned long timeout) throw (lang::Exception)
Waits for the Count to hit zero, or a timeout.
- virtual void **countDown** ()
Counts down the latch, releasing all waiting threads when the count hits zero.
- virtual int **getCount** () const
Gets the current count.

6.248.1 Constructor & Destructor Documentation

6.248.1.1 decaf::util::concurrent::CountDownLatch::CountDownLatch (int count)

Constructor.

Parameters

count - number to count down from.

6.248.1.2 `virtual decaf::util::concurrent::CountDownLatch::~~CountDownLatch () [virtual]`

6.248.2 Member Function Documentation

6.248.2.1 `virtual void decaf::util::concurrent::CountDownLatch::await () throw (lang::Exception) [virtual]`

Waits for the Count to be zero, and then returns.

Exceptions

Exception

6.248.2.2 `virtual bool decaf::util::concurrent::CountDownLatch::await (unsigned long timeOut) throw (lang::Exception) [virtual]`

Waits for the Count to hit zero, or a timeout.

Parameters

timeOut - time in milliseconds to wait.

Returns

true if the wait made it to count zero, otherwise false

6.248.2.3 `virtual void decaf::util::concurrent::CountDownLatch::countDown () [virtual]`

Counts down the latch, releasing all waiting threads when the count hits zero.

6.248.2.4 `virtual int decaf::util::concurrent::CountDownLatch::getCount () const [inline, virtual]`

Gets the current count.

Returns

int count value

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/CountDownLatch.h`

6.249 activemq::commands::DataArrayResponse Class Reference

```
#include <src/main/activemq/commands/DataArrayResponse.h>
```

Inheritance diagram for `activemq::commands::DataArrayResponse`:

Public Member Functions

- **DataArrayResponse** ()
- virtual **~DataArrayResponse** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **DataArrayResponse * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1372) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent.*
- virtual const std::vector< **decaf::lang::Pointer**< **DataStructure** > > & **getData** () const
- virtual std::vector< **decaf::lang::Pointer**< **DataStructure** > > & **getData** ()
- virtual void **setData** (const std::vector< **decaf::lang::Pointer**< **DataStructure** > > &data)

Static Public Attributes

- static const unsigned char **ID_DATAARRAYRESPONSE** = 33

Protected Member Functions

- **DataArrayResponse** (const **DataArrayResponse** &)
- **DataArrayResponse** & **operator=** (const **DataArrayResponse** &)

Protected Attributes

- std::vector< **decaf::lang::Pointer**< **DataStructure** > > **data**

6.249.1 Constructor & Destructor Documentation

6.249.1.1 `activemq::commands::DataArrayResponse::DataArrayResponse (const DataArrayResponse &) [inline, protected]`

6.249.1.2 `activemq::commands::DataArrayResponse::DataArrayResponse ()`

6.249.1.3 `virtual activemq::commands::DataArrayResponse::~~DataArrayResponse () [virtual]`

6.249.2 Member Function Documentation

6.249.2.1 `virtual DataArrayResponse* activemq::commands::DataArrayResponse::cloneDataStructure () const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Reimplemented from `activemq::commands::Response` (p.2612).

6.249.2.2 `virtual void activemq::commands::DataArrayResponse::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

Reimplemented from `activemq::commands::Response` (p.2612).

6.249.2.3 `virtual bool activemq::commands::DataArrayResponse::equals (const DataStructure * value) const [virtual]`

Compares the `DataStructure` (p.1372) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::Response` (p.2613).

- 6.249.2.4** `virtual std::vector< decaf::lang::Pointer<DataStructure> >& activemq::commands::DataArrayResponse::getData () [virtual]`
- 6.249.2.5** `virtual const std::vector< decaf::lang::Pointer<DataStructure> >& activemq::commands::DataArrayResponse::getData () const [virtual]`
- 6.249.2.6** `virtual unsigned char activemq::commands::DataArrayResponse::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataStructure** (p. 1372) type copy.

Reimplemented from **activemq::commands::Response** (p. 2613).

- 6.249.2.7** `DataArrayResponse& activemq::commands::DataArrayResponse::operator= (const DataArrayResponse &) [inline, protected]`
- 6.249.2.8** `virtual void activemq::commands::DataArrayResponse::setData (const std::vector< decaf::lang::Pointer< DataStructure > > & data) [virtual]`
- 6.249.2.9** `virtual std::string activemq::commands::DataArrayResponse::toString () const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1372) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::Response** (p. 2614).

6.249.3 Field Documentation

- 6.249.3.1** `std::vector< decaf::lang::Pointer<DataStructure> > activemq::commands::DataArrayResponse::data [protected]`
- 6.249.3.2** `const unsigned char activemq::commands::DataArrayResponse::ID_ - DATAARRAYRESPONSE = 33 [static]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/DataArrayResponse.h`

6.250 activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1271).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/DataArrayResponseMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller:

Public Member Functions

- **DataArrayResponseMarshaller** ()
- virtual **~DataArrayResponseMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.250.1 Detailed Description

Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1271). NOTE! This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.250.2 Constructor & Destructor Documentation

6.250.2.1 `activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller::DataArrayR
() [inline]`

6.250.2.2 `virtual
activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller::~~DataArrayR
() [inline, virtual]`

6.250.3 Member Function Documentation

6.250.3.1 `virtual commands::DataStructure* ac-
tivemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller::createObject
() const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 2625).

6.250.3.2 `virtual unsigned char ac-
tivemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller::getDataStruct
() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 2626).

6.250.3.3 `virtual void ac-
tivemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller::looseMarshal
(OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut) throw (
decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 2626).

6.250.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 2626).

6.250.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 2627).

```
6.250.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 2627).

```
6.250.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 2628).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/DataArrayResponseMarshaller.h`

6.251 activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1275).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/DataArrayResponseMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller:

Public Member Functions

- **DataArrayResponseMarshaller** ()
- virtual **~DataArrayResponseMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.251.1 Detailed Description

Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1275). NOTE! This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.251.2 Constructor & Destructor Documentation

6.251.2.1 `activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller::DataArrayR
() [inline]`

6.251.2.2 `virtual
activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller::~~DataArrayR
() [inline, virtual]`

6.251.3 Member Function Documentation

6.251.3.1 `virtual commands::DataStructure* ac-
tivemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller::createObject
() const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 2639).

6.251.3.2 `virtual unsigned char ac-
tivemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller::getDataStruct
() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 2639).

6.251.3.3 `virtual void ac-
tivemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller::looseMarshal
(OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut) throw (
decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 2639).

6.251.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 2640).

6.251.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 2640).

```

6.251.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 2641).

```

6.251.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]

```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 2641).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/DataArrayResponseMarshaller.h`

6.252 activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1279).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/DataArrayResponseMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller:

Public Member Functions

- **DataArrayResponseMarshaller** ()
- virtual **~DataArrayResponseMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.252.1 Detailed Description

Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1279). NOTE! This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.252.2 Constructor & Destructor Documentation

6.252.2.1 `activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller::DataArrayR
() [inline]`

6.252.2.2 `virtual
activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller::~~DataArrayR
() [inline, virtual]`

6.252.3 Member Function Documentation

6.252.3.1 `virtual commands::DataStructure* ac-
tivemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller::createObject
() const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 2634).

6.252.3.2 `virtual unsigned char ac-
tivemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller::getDataStruct
() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 2635).

6.252.3.3 `virtual void ac-
tivemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller::looseMarshal
(OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut) throw (
decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 2635).

6.252.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 2635).

6.252.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 2636).

```
6.252.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 2636).

```
6.252.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 2637).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/DataArrayResponseMarshaller.h`

6.253 activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1283).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/DataArrayResponseMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller:

Public Member Functions

- **DataArrayResponseMarshaller** ()
- virtual **~DataArrayResponseMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.253.1 Detailed Description

Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1283). NOTE! This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.253.2 Constructor & Destructor Documentation

6.253.2.1 `activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller::DataArrayR
() [inline]`

6.253.2.2 `virtual
activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller::~~DataArrayR
() [inline, virtual]`

6.253.3 Member Function Documentation

6.253.3.1 `virtual commands::DataStructure* ac-
tivemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller::createObject
() const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 2630).

6.253.3.2 `virtual unsigned char ac-
tivemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller::getDataStruct
() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 2630).

6.253.3.3 `virtual void ac-
tivemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller::looseMarshal
(OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut) throw (
decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 2630).

6.253.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 2631).

6.253.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 2631).

```

6.253.3.6 virtual void ac-
    tivemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller::tightMarshal2
    ( OpenWireFormat * wireFormat, commands::DataStructure
    * dataStructure, decaf::io::DataOutputStream * dataOut,
    utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 2632).

```

6.253.3.7 virtual void ac-
    tivemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller::tightUnmarshal
    ( OpenWireFormat * wireFormat, commands::DataStructure
    * dataStructure, decaf::io::DataInputStream * dataIn,
    utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]

```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 2632).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/DataArrayResponseMarshaller.h`

6.254 activemq::wireformat::openwire::marshal::v2::DataArrayResponse Class Reference

Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1287).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/DataArrayResponseMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller:

Public Member Functions

- **DataArrayResponseMarshaller** ()
- virtual **~DataArrayResponseMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.254.1 Detailed Description

Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1287). NOTE! This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.254.2 Constructor & Destructor Documentation

6.254.2.1 `activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller::DataArrayR
() [inline]`

6.254.2.2 `virtual
activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller::~~DataArrayR
() [inline, virtual]`

6.254.3 Member Function Documentation

6.254.3.1 `virtual commands::DataStructure* ac-
tivemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller::createObject
() const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 2621).

6.254.3.2 `virtual unsigned char ac-
tivemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller::getDataStruct
() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 2621).

6.254.3.3 `virtual void ac-
tivemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller::looseMarshal
(OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut) throw (
decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 2621).

6.254.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 2622).

6.254.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 2622).

```
6.254.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 2623).

```
6.254.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 2623).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/DataArrayResponseMarshaller.h`

6.255 decaf::io::DataInputStream Class Reference

A data input stream lets an application read primitive Java data types from an underlying input stream in a machine-independent way.

```
#include <src/main/decaf/io/DataInputStream.h>
```

Inheritance diagram for decaf::io::DataInputStream:

Public Member Functions

- **DataInputStream** (**InputStream** *inputStream, bool own=false)
*Creates a **DataInputStream** (p. 1291) that uses the specified underlying **InputStream** (p. 1630).*
- virtual ~**DataInputStream** ()
- virtual int **read** () throw (**IOException**)
Reads the next byte of data from this input stream.
- virtual int **read** (std::vector< unsigned char > &buffer) throw (**io::IOException**)
Reads some number of bytes from the contained input stream and stores them into the buffer array b.
- virtual int **read** (unsigned char *buffer, std::size_t offset, std::size_t length) throw (**io::IOException**, **lang::exceptions::NullPointerException**)
Reads up to len bytes of data from the contained input stream into an array of bytes.
- virtual bool **readBoolean** () throw (**io::IOException**, **io::EOFException**)
Reads one input byte and returns true if that byte is nonzero, false if that byte is zero.
- virtual char **readByte** () throw (**io::IOException**, **io::EOFException**)
Reads and returns one input byte.
- virtual unsigned char **readUnsignedByte** () throw (**io::IOException**, **io::EOFException**)
Reads one input byte, zero-extends it to type int, and returns the result, which is therefore in the range 0 through 255.
- virtual char **readChar** () throw (**io::IOException**, **io::EOFException**)
Reads an input char and returns the char value.
- virtual double **readDouble** () throw (**io::IOException**, **io::EOFException**)
Reads eight input bytes and returns a double value.
- virtual float **readFloat** () throw (**io::IOException**, **io::EOFException**)
Reads four input bytes and returns a float value.
- virtual int **readInt** () throw (**io::IOException**, **io::EOFException**)
Reads four input bytes and returns an int value.

- virtual long long **readLong** () throw (io::IOException, io::EOFException)
Reads eight input bytes and returns a long value.
- virtual short **readShort** () throw (io::IOException, io::EOFException)
Reads two input bytes and returns a short value.
- virtual unsigned short **readUnsignedShort** () throw (io::IOException, io::EOFException)
Reads two input bytes and returns an int value in the range 0 through 65535.
- virtual std::string **readString** () throw (io::IOException, io::EOFException)
Reads an null terminated ASCII string to the stream and returns the string to the caller.
- virtual std::string **readUTF** () throw (io::IOException, io::EOFException, io::UTFDataFormatException)
Reads a modified UTF-8 encoded string in ASCII format and returns it, this is only useful if you know for sure that the string that is to be read was a string that contained all ASCII values (0-255), if so this method will throw a UTFFormatException.
- virtual void **readFully** (std::vector< unsigned char > &buffer) throw (io::IOException, io::EOFException)
Reads some bytes from an input stream and stores them into the buffer array buffer.
- virtual void **readFully** (unsigned char *buffer, std::size_t offset, std::size_t length) throw (io::IOException, io::EOFException, lang::exceptions::NullPointerException)
Reads length bytes from an input stream.
- virtual std::size_t **skip** (std::size_t num) throw (io::IOException, lang::exceptions::UnsupportedOperationException)
Makes an attempt to skip over n bytes of data from the input stream, discarding the skipped bytes.

6.255.1 Detailed Description

A data input stream lets an application read primitive Java data types from an underlying input stream in a machine-independent way. An application uses a data output stream to write data that can later be read by a data input stream.

Due to the lack of garbage collection in C++ a design decision was made to add a boolean parameter to the constructor indicating if the wrapped **InputStream** (p. 1630) is owned by this object. That way creation of the underlying stream can occur in a Java like way. Ex:

```
DataInputStream (p. 1291) os = new DataInputStream (p. 1291)( new InputStream(), true )
```

Since

1.0

6.255.2 Constructor & Destructor Documentation

6.255.2.1 decaf::io::DataInputStream::DataInputStream (*InputStream* * *inputStream*, *bool* *own* = *false*)

Creates a **DataInputStream** (p. 1291) that uses the specified underlying **InputStream** (p. 1630).

Parameters

inputStream the **InputStream** (p. 1630) instance to wrap.

own indicates if this class owns the wrapped string defaults to false.

6.255.2.2 virtual decaf::io::DataInputStream::~~DataInputStream () [virtual]

6.255.3 Member Function Documentation

6.255.3.1 virtual int decaf::io::DataInputStream::read () throw (*IOException*) [inline, virtual]

Reads the next byte of data from this input stream.

The value byte is returned as an unsigned char in the range 0 to 255. If no byte is available because the end of the stream has been reached, the value -1 is returned. This method blocks until input data is available, the end of the stream is detected, or an exception is thrown. This method simply performs `in.read()` and returns the result.

Returns

The next byte.

Exceptions

IOException (p. 1707) thrown if an error occurs.

Reimplemented from **decaf::io::FilterInputStream** (p. 1532).

References `decaf::io::FilterInputStream::read()`.

6.255.3.2 virtual int decaf::io::DataInputStream::read (*std::vector< unsigned char>* & *buffer*) throw (*io::IOException*) [virtual]

Reads some number of bytes from the contained input stream and stores them into the buffer array *b*.

The number of bytes actually read is returned as an integer. This method blocks until input data is available, end of file is detected, or an exception is thrown.

If the length of *buffer* is zero, then no bytes are read and 0 is returned; otherwise, there is an attempt to read at least one byte. If no byte is available because the stream is at end of file, the value -1 is returned; otherwise, at least one byte is read and stored into *buffer*.

The first byte read is stored into element *buffer*[0], the next one into *buffer*[1], and so on. The number of bytes read is, at most, equal to the length of *buffer*. Let *k* be the number of bytes actually read; these bytes will be stored in elements *b*[0] through *b*[*k*-1], leaving elements *buffer*[*k*] through *buffer*[*buffer.length*-1] unaffected.

If the first byte cannot be read for any reason other than end of file, then an **IOException** (p.1707) is thrown. In particular, an **IOException** (p.1707) is thrown if the input stream has been closed.

The `read(buffer)` method has the same effect as: `read(buffer, 0, b.length)`

Parameters

buffer - byte array to insert read data into

Returns

the total number of bytes read, or -1 if there is no more data because the stream is EOF.

Exceptions

IOException (p. 1707)

6.255.3.3 `virtual int decaf::io::DataInputStream::read (unsigned char * buffer,
std::size_t offset, std::size_t length) throw (io::IOException,
lang::exceptions::NullPointerException) [virtual]`

Reads up to len bytes of data from the contained input stream into an array of bytes.

An attempt is made to read as many as len bytes, but a smaller number may be read, possibly zero. The number of bytes actually read is returned as an integer.

This method blocks until input data is available, end of file is detected, or an exception is thrown.

If buffer is null, a NullPointerException is thrown.

If off is negative, or len is negative then an IndexOutOfBoundsException is thrown, if off + len is greater than the allocated length of the array, an **IOException** (p.1707) will result depending on the platform and compiler settings.

If len is zero, then no bytes are read and 0 is returned; otherwise, there is an attempt to read at least one byte. If no byte is available because the stream is at end of file, the value -1 is returned; otherwise, at least one byte is read and stored into buffer.

The first byte read is stored into element b[off], the next one into buffer[off+1], and so on. The number of bytes read is, at most, equal to len. Let k be the number of bytes actually read; these bytes will be stored in elements buffer[off] through buffer[off+k-1], leaving elements buffer[off+k] through buffer[off+len-1] unaffected.

In every case, elements buffer[0] through buffer[off] and elements buffer[off+len] through buffer[buffer.length-1] are unaffected.

If the first byte cannot be read for any reason other than end of file, then an **IOException** (p.1707) is thrown. In particular, an **IOException** (p.1707) is thrown if the input stream has been closed.

Parameters

buffer - byte array to insert read data into

offset - location in buffer to start writing

length - number of bytes to read

Returns

the total number of bytes read, or -1 if there is no more data because the stream is EOF.

Exceptions

IOException (p. 1707)

NullPointerException if the buffer is null

Reimplemented from **decaf::io::FilterInputStream** (p. 1533).

6.255.3.4 `virtual bool decaf::io::DataInputStream::readBoolean () throw (io::IOException, io::EOFException) [virtual]`

Reads one input byte and returns true if that byte is nonzero, false if that byte is zero.

Returns

the boolean value read.

Exceptions

IOException (p. 1707)

EOFException (p. 1474)

6.255.3.5 `virtual char decaf::io::DataInputStream::readByte () throw (io::IOException, io::EOFException) [virtual]`

Reads and returns one input byte.

The byte is treated as a signed value in the range -128 through 127, inclusive.

Returns

the 8-bit value read.

Exceptions

IOException (p. 1707)

EOFException (p. 1474)

6.255.3.6 `virtual char decaf::io::DataInputStream::readChar () throw (io::IOException, io::EOFException) [virtual]`

Reads an input char and returns the char value.

A ascii char is made up of one bytes. This returns the same result as **readByte**

Returns

the 8 bit char read

Exceptions

IOException (p. 1707)

EOFException (p. 1474)

6.255.3.7 virtual double decaf::io::DataInputStream::readDouble () throw (io::IOException, io::EOFException) [virtual]

Reads eight input bytes and returns a double value.

It does this by first constructing a long long value in exactly the manner of the readlong method, then converting this long value to a double in exactly the manner of the method Double.longBitsToDouble.

Returns

the double value read

Exceptions

IOException (p. 1707)

EOFException (p. 1474)

6.255.3.8 virtual float decaf::io::DataInputStream::readFloat () throw (io::IOException, io::EOFException) [virtual]

Reads four input bytes and returns a float value.

It does this by first constructing an int value in exactly the manner of the readInt method, then converting this int value to a float in exactly the manner of the method Float.intBitsToFloat.

Returns

the float value read

Exceptions

IOException (p. 1707)

EOFException (p. 1474)

6.255.3.9 virtual void decaf::io::DataInputStream::readFully (std::vector< unsigned char > & buffer) throw (io::IOException, io::EOFException) [virtual]

Reads some bytes from an input stream and stores them into the buffer array buffer.

The number of bytes read is equal to the length of buffer.

This method blocks until one of the following conditions occurs: * buffer.size() bytes of input data are available, in which case a normal return is made. * End of file is detected, in which case an **EOFException** (p. 1474) is thrown. * An I/O error occurs, in which case an **IOException** (p. 1707) other than **EOFException** (p. 1474) is thrown.

If buffer.size() is zero, then no bytes are read. Otherwise, the first byte read is stored into element b[0], the next one into buffer[1], and so on. If an exception is thrown from this method, then it may be that some but not all bytes of buffer have been updated with data from the input stream.

Parameters

buffer - vector of char that is read to its size()

Exceptions***IOException*** (p. 1707)***EOFException*** (p. 1474)

6.255.3.10 virtual void decaf::io::DataInputStream::readFully (unsigned char * *buffer*, std::size_t *offset*, std::size_t *length*) throw (io::IOException, io::EOFException, lang::exceptions::NullPointerException) [virtual]

Reads length bytes from an input stream.

This method blocks until one of the following conditions occurs: * length bytes of input data are available, in which case a normal return is made. * End of file is detected, in which case an **EOFException** (p. 1474) is thrown. * An I/O error occurs, in which case an **IOException** (p. 1707) other than **EOFException** (p. 1474) is thrown.

If buffer is null, a NullPointerException is thrown. If offset is negative, or len is negative, or offset+length is greater than the length of the array buffer, then an IndexOutOfBoundsException is thrown. If len is zero, then no bytes are read. Otherwise, the first byte read is stored into element buffer[off], the next one into buffer[offset+1], and so on. The number of bytes read is, at most, equal to len.

Parameters

buffer - byte array to insert read data into

offset - location in buffer to start writing

length - number of bytes to read

Exceptions***IOException*** (p. 1707)***EOFException*** (p. 1474)***NullPointerException*** if the buffer is null

6.255.3.11 virtual int decaf::io::DataInputStream::readInt () throw (io::IOException, io::EOFException) [virtual]

Reads four input bytes and returns an int value.

Let a be the first byte read, b be the second byte, c be the third byte, and d be the fourth byte. The value returned is:

$$(((a \& 0xff) << 24) | ((b \& 0xff) << 16) | ((c \& 0xff) << 8) | (d \& 0xff))$$
Returns

the int value read

Exceptions***IOException*** (p. 1707)***EOFException*** (p. 1474)

6.255.3.12 virtual long long decaf::io::DataInputStream::readLong () throw (io::IOException, io::EOFException) [virtual]

Reads eight input bytes and returns a long value.

Let a be the first byte read, b be the second byte, c be the third byte, d be the fourth byte, e be the fifth byte, f be the sixth byte, g be the seventh byte, and h be the eighth byte. The value returned is: (((long)(a & 0xff) << 56) | ((long)(b & 0xff) << 48) | ((long)(c & 0xff) << 40) | ((long)(d & 0xff) << 32) | ((long)(e & 0xff) << 24) | ((long)(f & 0xff) << 16) | ((long)(g & 0xff) << 8) | ((long)(h & 0xff)))

Returns

the 64 bit long long read

Exceptions

IOException (p. 1707)

EOFException (p. 1474)

6.255.3.13 virtual short decaf::io::DataInputStream::readShort () throw (io::IOException, io::EOFException) [virtual]

Reads two input bytes and returns a short value.

Let a be the first byte read and b be the second byte. The value returned is: (short)((a << 8) | (b & 0xff))

Returns

the 16 bit short value read

Exceptions

IOException (p. 1707)

EOFException (p. 1474)

6.255.3.14 virtual std::string decaf::io::DataInputStream::readString () throw (io::IOException, io::EOFException) [virtual]

Reads an null terminated ASCII string to the stream and returns the string to the caller.

Returns

string object containing the string read.

Exceptions

IOException (p. 1707)

EOFException (p. 1474)

6.255.3.15 virtual unsigned char decaf::io::DataInputStream::readUnsignedByte () throw (io::IOException, io::EOFException) [virtual]

Reads one input byte, zero-extends it to type int, and returns the result, which is therefore in the range 0 through 255.

Returns

the 8 bit unsigned value read

Exceptions

IOException (p. 1707)

EOFException (p. 1474)

6.255.3.16 virtual unsigned short decaf::io::DataInputStream::readUnsignedShort () throw (io::IOException, io::EOFException) [virtual]

Reads two input bytes and returns an int value in the range 0 through 65535.

Let a be the first byte read and b be the second byte. The value returned is: (((a & 0xff) << 8) | (b & 0xff))

Returns

the 16 bit unsigned short read

Exceptions

IOException (p. 1707)

EOFException (p. 1474)

6.255.3.17 virtual std::string decaf::io::DataInputStream::readUTF () throw (io::IOException, io::EOFException, io::UTFDataFormatException) [virtual]

Reads a modified UTF-8 encoded string in ASCII format and returns it, this is only useful if you know for sure that the string that is to be read was a string that contained all ASCII values (0-255), if so this method will throw a UTFFormatException.

This method reads String value written from a Java **DataOutputStream** (p. 1300) and assumes that the length prefix the precedes the encoded UTF-8 bytes is an unsigned short, which implies that the String will be no longer than 65535 characters.

Returns

The decoded string read from stream.

Exceptions

IOException (p. 1707)

EOFException (p. 1474)

UTFDataFormatException (p. 3139)

6.255.3.18 `virtual std::size_t decaf::io::DataInputStream::skip
(std::size_t num) throw (io::IOException,
lang::exceptions::UnsupportedOperationException) [virtual]`

Makes an attempt to skip over *n* bytes of data from the input stream, discarding the skipped bytes.

However, it may skip over some smaller number of bytes, possibly zero. This may result from any of a number of conditions; reaching end of file before *n* bytes have been skipped is only one possibility. This method never throws an **EOFException** (p.1474). The actual number of bytes skipped is returned.

Parameters

num - number of bytes to skip

Returns

the total number of bytes skipped

Reimplemented from **decaf::io::FilterInputStream** (p.1534).

The documentation for this class was generated from the following file:

- `src/main/decaf/io/DataInputStream.h`

6.256 decaf::io::DataOutputStream Class Reference

A data output stream lets an application write primitive Java data types to an output stream in a portable way.

`#include <src/main/decaf/io/DataOutputStream.h>`

Inheritance diagram for `decaf::io::DataOutputStream`:

Public Member Functions

- **DataOutputStream** (`OutputStream *outputStream`, `bool own=false`)
Creates a new data output stream to write data to the specified underlying output stream.
- `virtual ~DataOutputStream ()`
- `virtual std::size_t size () const`
Returns the current value of the counter written, the number of bytes written to this data output stream so far.
- `virtual void write (unsigned char c) throw (IOException)`
Writes a single byte to the output stream.
- `virtual void write (const unsigned char *buffer, std::size_t offset, std::size_t len) throw (IOException, lang::exceptions::NullPointerException)`
Writes an array of bytes to the output stream.

- virtual void **write** (const std::vector< unsigned char > &**buffer**) throw (IOException)
Writes an array of bytes to the output stream.
- virtual void **writeBoolean** (bool value) throw (IOException)
Writes a boolean to the underlying output stream as a 1-byte value.
- virtual void **writeByte** (unsigned char value) throw (IOException)
Writes out a byte to the underlying output stream as a 1-byte value.
- virtual void **writeShort** (short value) throw (IOException)
Writes a short to the underlying output stream as two bytes, high byte first.
- virtual void **writeUnsignedShort** (unsigned short value) throw (IOException)
Writes a unsigned short to the bytes message stream as a 2 byte value.
- virtual void **writeChar** (char value) throw (IOException)
Writes out a char to the underlying output stream as a one byte value If no exception is thrown, the counter written is incremented by 1.
- virtual void **writeInt** (int value) throw (IOException)
Writes an int to the underlying output stream as four bytes, high byte first.
- virtual void **writeLong** (long long value) throw (IOException)
Writes an 64 bit long to the underlying output stream as eight bytes, high byte first.
- virtual void **writeFloat** (float value) throw (IOException)
Converts the float argument to an int using the floatToIntBits method in class Float, and then writes that int value to the underlying output stream as a 4-byte quantity, high byte first.
- virtual void **writeDouble** (double value) throw (IOException)
Converts the double argument to a long using the doubleToLongBits method in class Double, and then writes that long value to the underlying output stream as an 8-byte quantity, high byte first.
- virtual void **writeBytes** (const std::string &value) throw (IOException)
Writes out the string to the underlying output stream as a sequence of bytes.
- virtual void **writeChars** (const std::string &value) throw (IOException)
Writes a string to the underlying output stream as a sequence of characters.
- virtual void **writeUTF** (const std::string &value) throw (IOException, UTFDataFormatException)
Writes out the string to the underlying output stream as a modeified UTF-8 encoded sequence of bytes.

Protected Attributes

- std::size_t **written**
- unsigned char **buffer** [8]

6.256.1 Detailed Description

A data output stream lets an application write primitive Java data types to an output stream in a portable way. An application can then use a data input stream to read the data back in.

6.256.2 Constructor & Destructor Documentation

6.256.2.1 `decaf::io::DataOutputStream::DataOutputStream (OutputStream * outputStream, bool own = false)`

Creates a new data output stream to write data to the specified underlying output stream.

Parameters

outputStream a stream to wrap with this one.

own true if this objects owns the stream that it wraps.

6.256.2.2 `virtual decaf::io::DataOutputStream::~~DataOutputStream ()` [virtual]

6.256.3 Member Function Documentation

6.256.3.1 `virtual std::size_t decaf::io::DataOutputStream::size () const` [inline, virtual]

Returns the current value of the counter written, the number of bytes written to this data output stream so far.

If the counter overflows, it will be wrapped to Integer.MAX_VALUE.

Returns

the value of the written field.

6.256.3.2 `virtual void decaf::io::DataOutputStream::write (unsigned char c)` `throw (IOException)` [virtual]

Writes a single byte to the output stream.

If no exception is thrown, the counter written is incremented by 1.

Parameters

c the byte.

Exceptions

IOException (p. 1707) thrown if an error occurs.

Reimplemented from `decaf::io::FilterOutputStream` (p. 1542).

6.256.3.3 virtual void decaf::io::DataOutputStream::write (const std::vector< unsigned char > & *buffer*) throw (IOException) [virtual]

Writes an array of bytes to the output stream.

Parameters

buffer The bytes to write.

Exceptions

IOException (p. 1707) thrown if an error occurs.

Reimplemented from **decaf::io::FilterOutputStream** (p. 1542).

6.256.3.4 virtual void decaf::io::DataOutputStream::write (const unsigned char * *buffer*, std::size_t *offset*, std::size_t *len*) throw (IOException, lang::exceptions::NullPointerException) [virtual]

Writes an array of bytes to the output stream.

the counter written is incremented by len.

Parameters

buffer The array of bytes to write.

offset the position in buffer to start writing from.

len The number of bytes from the buffer to be written.

Exceptions

IOException (p. 1707) thrown if an error occurs.

NullPointerException if buffer is Null

Reimplemented from **decaf::io::FilterOutputStream** (p. 1543).

6.256.3.5 virtual void decaf::io::DataOutputStream::writeBoolean (bool *value*) throw (IOException) [virtual]

Writes a boolean to the underlying output stream as a 1-byte value.

The value true is written out as the value (byte)1; the value false is written out as the value (byte)0. If no exception is thrown, the counter written is incremented by 1.

Parameters

value the boolean to write.

Exceptions

IOException (p. 1707)

6.256.3.6 `virtual void decaf::io::DataOutputStream::writeByte (unsigned char value) throw (IOException) [virtual]`

Writes out a byte to the underlying output stream as a 1-byte value.

If no exception is thrown, the counter written is incremented by 1.

Parameters

value the unsigned char value to write.

Exceptions

IOException (p. 1707)

6.256.3.7 `virtual void decaf::io::DataOutputStream::writeBytes (const std::string & value) throw (IOException) [virtual]`

Writes out the string to the underlying output stream as a sequence of bytes.

Each character in the string is written out, in sequence, by discarding its high eight bits. If no exception is thrown, the counter written is incremented by the length of value. The value written does not include a trailing null as that is not part of the sequence of bytes, if the null is needed, then use the writeChars method.

Parameters

value the value to write.

Exceptions

IOException (p. 1707)

6.256.3.8 `virtual void decaf::io::DataOutputStream::writeChar (char value) throw (IOException) [virtual]`

Writes out a char to the underlying output stream as a one byte value. If no exception is thrown, the counter written is incremented by 1.

Parameters

value the value to write.

Exceptions

IOException (p. 1707)

6.256.3.9 `virtual void decaf::io::DataOutputStream::writeChars (const std::string & value) throw (IOException) [virtual]`

Writes a string to the underlying output stream as a sequence of characters.

Each character is written to the data output stream as if by the writeChar method. If no exception is thrown, the counter written is incremented by the length of value. The trailing NULL character is written by this method.

Parameters

value the value to write.

Exceptions

IOException (p. 1707)

6.256.3.10 virtual void decaf::io::DataOutputStream::writeDouble (double *value*) throw (IOException) [virtual]

Converts the double argument to a long using the doubleToLongBits method in class Double, and then writes that long value to the underlying output stream as an 8-byte quantity, high byte first.

If no exception is thrown, the counter written is incremented by 8.

Parameters

value the value to write.

Exceptions

IOException (p. 1707)

6.256.3.11 virtual void decaf::io::DataOutputStream::writeFloat (float *value*) throw (IOException) [virtual]

Converts the float argument to an int using the floatToIntBits method in class Float, and then writes that int value to the underlying output stream as a 4-byte quantity, high byte first.

If no exception is thrown, the counter written is incremented by 4.

Parameters

value the value to write.

Exceptions

IOException (p. 1707)

6.256.3.12 virtual void decaf::io::DataOutputStream::writeInt (int *value*) throw (IOException) [virtual]

Writes an int to the underlying output stream as four bytes, high byte first.

If no exception is thrown, the counter written is incremented by 4.

Parameters

value the value to write.

Exceptions

IOException (p. 1707)

6.256.3.13 `virtual void decaf::io::DataOutputStream::writeLong (long long value) throw (IOException)` [virtual]

Writes an 64 bit long to the underlying output stream as eight bytes, high byte first.

If no exception is thrown, the counter written is incremented by 8.

Parameters

value the value to write.

Exceptions

IOException (p. 1707)

6.256.3.14 `virtual void decaf::io::DataOutputStream::writeShort (short value) throw (IOException)` [virtual]

Writes a short to the underlying output stream as two bytes, high byte first.

If no exception is thrown, the counter written is incremented by 2.

Parameters

value the value to write.

Exceptions

IOException (p. 1707)

6.256.3.15 `virtual void decaf::io::DataOutputStream::writeUnsignedShort (unsigned short value) throw (IOException)` [virtual]

Writes a unsigned short to the bytes message stream as a 2 byte value.

Parameters

value - unsigned short to write to the stream

Exceptions

IOException (p. 1707)

6.256.3.16 `virtual void decaf::io::DataOutputStream::writeUTF (const std::string & value) throw (IOException, UTFDataFormatException)` [virtual]

Writes out the string to the underlying output stream as a modified UTF-8 encoded sequence of bytes.

The first two bytes written are indicate its encoded length followed by the rest of the string's characters encoded as modified UTF-8. The length represent the encoded length of the data not the actual length of the string.

Parameters

value the value to write.

Exceptions

IOException (p. 1707) - on a write error

UTFDataFormatException (p. 3139) - if encoded size if greater than 65535

6.256.4 Field Documentation

6.256.4.1 unsigned char decaf::io::DataOutputStream::buffer[8] [protected]

6.256.4.2 std::size_t decaf::io::DataOutputStream::written [protected]

The documentation for this class was generated from the following file:

- src/main/decaf/io/**DataOutputStream.h**

6.257 activemq::commands::DataResponse Class Reference

```
#include <src/main/activemq/commands/DataResponse.h>
```

Inheritance diagram for activemq::commands::DataResponse:

Public Member Functions

- **DataResponse** ()
- virtual **~DataResponse** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **DataResponse * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1372) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **DataStructure** > &**getData** () const
- virtual **Pointer**< **DataStructure** > &**getData** ()
- virtual void **setData** (const **Pointer**< **DataStructure** > &data)

Static Public Attributes

- static const unsigned char **ID_DATARESPONSE** = 32

Protected Member Functions

- **DataResponse** (const **DataResponse** &)
- **DataResponse** & **operator=** (const **DataResponse** &)

Protected Attributes

- **Pointer< DataStructure > data**

6.257.1 Constructor & Destructor Documentation

6.257.1.1 **activemq::commands::DataResponse::DataResponse** (const **DataResponse** &) [inline, protected]

6.257.1.2 **activemq::commands::DataResponse::DataResponse** ()

6.257.1.3 **virtual activemq::commands::DataResponse::~~DataResponse** () [virtual]

6.257.2 Member Function Documentation

6.257.2.1 **virtual DataResponse* activemq::commands::DataResponse::cloneDataStructure** () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Reimplemented from **activemq::commands::Response** (p. 2612).

6.257.2.2 **virtual void activemq::commands::DataResponse::copyDataStructure** (const **DataStructure** * *src*) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

Reimplemented from **activemq::commands::Response** (p. 2612).

6.257.2.3 virtual bool activemq::commands::DataResponse::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::Response** (p. 2613).

6.257.2.4 virtual Pointer<DataStructure>& activemq::commands::DataResponse::getData () [virtual]**6.257.2.5 virtual const Pointer<DataStructure>& activemq::commands::DataResponse::getData () const** [virtual]**6.257.2.6 virtual unsigned char activemq::commands::DataResponse::getDataStructureType () const** [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataStructure** (p. 1372) type copy.

Reimplemented from **activemq::commands::Response** (p. 2613).

6.257.2.7 DataResponse& activemq::commands::DataResponse::operator= (const DataResponse &) [inline, protected]**6.257.2.8 virtual void activemq::commands::DataResponse::setData (const Pointer< DataStructure > & *data*)** [virtual]**6.257.2.9 virtual std::string activemq::commands::DataResponse::toString () const** [virtual]

Returns a string containing the information for this **DataStructure** (p. 1372) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::Response** (p. 2614).

6.257.3 Field Documentation

6.257.3.1 `Pointer<DataStructure> activemq::commands::DataResponse::data`
[protected]

6.257.3.2 `const unsigned char activemq::commands::DataResponse::ID_ -
DATARESPONSE = 32` [static]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/DataResponse.h`

6.258 `activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller` Class Reference

Marshaling code for Open Wire Format for `DataResponseMarshaller` (p. 1310).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/DataResponseMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller`:

Public Member Functions

- `DataResponseMarshaller ()`
- `virtual ~DataResponseMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaler.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write a object instance to data output stream.
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)`

Un-marshall an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.258.1 Detailed Description

Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1310). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.258.2 Constructor & Destructor Documentation

6.258.2.1 **activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller::DataResponseMar**
() [inline]

6.258.2.2 **virtual**
activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller::~~DataResponseM
() [inline, virtual]

6.258.3 Member Function Documentation

6.258.3.1 **virtual commands::DataStructure* ac-**
tivemq::wireformat::openwire::marshal::v3::DataResponseMarshaller::createObject
() const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 2625).

6.258.3.2 **virtual unsigned char ac-**
tivemq::wireformat::openwire::marshal::v3::DataResponseMarshaller::getDataStructureTy
() const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 2626).

6.258.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 2626).

6.258.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 2626).

6.258.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 2627).

```
6.258.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::DataResponseMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 2627).

```
6.258.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::DataResponseMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 2628).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/DataResponseMarshaller.h`

6.259 activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1314).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/DataResponseMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller**:

Public Member Functions

- **DataResponseMarshaller** ()
- virtual **~DataResponseMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.259.1 Detailed Description

Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1314). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.259.2 Constructor & Destructor Documentation

6.259.2.1 activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller::DataResponseMarshaller () [inline]

6.259.2.2 virtual
activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller::~~DataResponseMarshaller () [inline, virtual]

6.259.3 Member Function Documentation

6.259.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 2634).

6.259.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 2635).

6.259.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 2635).

6.259.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 2635).

6.259.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 2636).

```
6.259.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::DataResponseMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 2636).

```
6.259.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::DataResponseMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 2637).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/DataResponseMarshaller.h`

6.260 activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1318).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/DataResponseMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller**:

Public Member Functions

- **DataResponseMarshaller** ()
- virtual **~DataResponseMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.260.1 Detailed Description

Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1318). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.260.2 Constructor & Destructor Documentation

6.260.2.1 activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller::DataResponseMarshaller () [inline]

6.260.2.2 virtual
activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller::~~DataResponseMarshaller () [inline, virtual]

6.260.3 Member Function Documentation

6.260.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 2630).

6.260.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 2630).

6.260.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 2630).

6.260.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 2631).

6.260.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 2631).

```
6.260.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::DataResponseMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 2632).

```
6.260.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::DataResponseMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 2632).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/DataResponseMarshaller.h`

6.261 **activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller** Class Reference

Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1322).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/DataResponseMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller**:

Public Member Functions

- **DataResponseMarshaller** ()
- virtual **~DataResponseMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.261.1 Detailed Description

Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1322). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.261.2 Constructor & Destructor Documentation

6.261.2.1 activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller::DataResponseMarshaller () [inline]

6.261.2.2 virtual
activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller::~~DataResponseMarshaller () [inline, virtual]

6.261.3 Member Function Documentation

6.261.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 2639).

6.261.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 2639).

6.261.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 2639).

6.261.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 2640).

6.261.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 2640).

```
6.261.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::DataResponseMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 2641).

```
6.261.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::DataResponseMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 2641).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/DataResponseMarshaller.h`

6.262 activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1326).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/DataResponseMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller**:

Public Member Functions

- **DataResponseMarshaller** ()
- virtual **~DataResponseMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.262.1 Detailed Description

Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1326). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.262.2 Constructor & Destructor Documentation

6.262.2.1 activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller::DataResponseMarshaller () [inline]

6.262.2.2 virtual
activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller::~~DataResponseMarshaller () [inline, virtual]

6.262.3 Member Function Documentation

6.262.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 2621).

6.262.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 2621).

6.262.3.3 virtual void activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 2621).

6.262.3.4 virtual void activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 2622).

6.262.3.5 virtual int activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 2622).

```
6.262.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::DataResponseMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 2623).

```
6.262.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::DataResponseMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 2623).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/DataResponseMarshaller.h`

6.263 activemq::wireformat::openwire::marshal::DataStreamMarshaller Class Reference

Base class for all classes that marshal commands for Openwire.

```
#include <src/main/activemq/wireformat/openwire/marshal/DataStreamMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::DataStreamMarshaller`:

Public Member Functions

- virtual `~DataStreamMarshaller ()`
- virtual unsigned char `getDataStructureType () const =0`
Gets the DataStructureType that this class marshals/unmarshals.
- virtual `commands::DataStructure * createObject () const =0`
Creates a new instance of the class that this class is a marshaling director for.
- virtual int `tightMarshal (OpenWireFormat *format, commands::DataStructure *command, utils::BooleanStream *bs)=0 throw (decaf::io::IOException)`
Tight Marshal to the given stream.
- virtual void `tightMarshal2 (OpenWireFormat *format, commands::DataStructure *command, decaf::io::DataOutputStream *ds, utils::BooleanStream *bs)=0 throw (decaf::io::IOException)`
Tight Marshal to the given stream.
- virtual void `tightUnmarshal (OpenWireFormat *format, commands::DataStructure *command, decaf::io::DataInputStream *dis, utils::BooleanStream *bs)=0 throw (decaf::io::IOException)`
Tight Un-marshal to the given stream.
- virtual void `looseMarshal (OpenWireFormat *format, commands::DataStructure *command, decaf::io::DataOutputStream *ds)=0 throw (decaf::io::IOException)`
Tight Marshal to the given stream.
- virtual void `looseUnmarshal (OpenWireFormat *format, commands::DataStructure *command, decaf::io::DataInputStream *dis)=0 throw (decaf::io::IOException)`
Loose Un-marshal to the given stream.

6.263.1 Detailed Description

Base class for all classes that marshal commands for Openwire.

6.263.2 Constructor & Destructor Documentation

6.263.2.1 virtual

activemq::wireformat::openwire::marshal::DataStreamMarshaller::~~DataStreamMarshaller
() [inline, virtual]

6.263.3 Member Function Documentation

6.263.3.1 virtual commands::DataStructure* ac-

tivemq::wireformat::openwire::marshal::DataStreamMarshaller::createObject
() const [pure virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns

newly allocated Command

Implemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller` (p. 152), `activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller` (p. 187), `activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller` (p. 290), `activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller` (p. 312), `activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller` (p. 352), `activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller` (p. 388), `activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller` (p. 441), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller` (p. 486), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller` (p. 510), `activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller` (p. 535), `activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller` (p. 558), `activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller` (p. 699), `activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller` (p. 727), `activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller` (p. 1065), `activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller` (p. 1088), `activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller` (p. 1114), `activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller` (p. 1144), `activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller` (p. 1180), `activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller` (p. 1204), `activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller` (p. 1228), `activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller` (p. 1255), `activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller` (p. 1280), `activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller` (p. 1315), `activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller` (p. 1408), `activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller` (p. 1437), `activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller` (p. 1492), `activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller` (p. 1581), `activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller` (p. 1675), `activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller` (p. 1735), `activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller` (p. 1752),

activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller (p. 1783),
 activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller (p. 1806),
 activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller (p. 1833),
 activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller
 (p. 1848), activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller
 (p. 1891), activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller
 (p. 2077), activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller
 (p. 2109), activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller
 (p. 2133), activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller (p. 2163),
 activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller (p. 2216), ac-
 tivemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller (p. 2263),
 activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller (p. 2336),
 activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller (p. 2441),
 activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller (p. 2463),
 activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller (p. 2491),
 activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller (p. 2541), ac-
 tivemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller
 (p. 2573), activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller
 (p. 2604), activemq::wireformat::openwire::marshal::v1::ResponseMarshaller (p. 2634),
 activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller (p. 2687), ac-
 tivemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller (p. 2710), ac-
 tivemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller (p. 2761), ac-
 tivemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller (p. 2912),
 activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller (p. 3050),
 activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller (p. 3172),
 activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller (p. 3210),
 activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller
 (p. 164), activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller
 (p. 199), activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller
 (p. 302), activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller
 (p. 324), activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller
 (p. 364), activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller
 (p. 400), activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller
 (p. 453), activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller
 (p. 498), activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller
 (p. 522), activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller
 (p. 547), activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller
 (p. 570), activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller (p. 711),
 activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller (p. 739), ac-
 tivemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller (p. 1077),
 activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller (p. 1100),
 activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller (p. 1126),
 activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller (p. 1152),
 activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller (p. 1188),
 activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller (p. 1212),
 activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller (p. 1240), ac-
 tivemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller (p. 1263), ac-
 tivemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller (p. 1288),
 activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller (p. 1327),
 activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller (p. 1396),
 activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller (p. 1421), ac-
 tivemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller (p. 1488),
 activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller (p. 1577),
 activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller (p. 1671),

activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller (p. 1723),
 activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller (p. 1748),
 activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller (p. 1771),
 activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller (p. 1794),
 activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller (p. 1817),
 activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller
 (p. 1844),
 activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller
 (p. 1879),
 activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller
 (p. 2061),
 activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller
 (p. 2097),
 activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller
 (p. 2121),
 activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller (p. 2147),
 activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller (p. 2208),
 activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller (p. 2251),
 activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller (p. 2323),
 activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller (p. 2425),
 activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller (p. 2451),
 activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller (p. 2475),
 activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller (p. 2549),
 activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller
 (p. 2569),
 activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller
 (p. 2588),
 activemq::wireformat::openwire::marshal::v2::ResponseMarshaller (p. 2621),
 activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller (p. 2691),
 activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller (p. 2706),
 activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller (p. 2773),
 activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller (p. 2928),
 activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller (p. 3058),
 activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller (p. 3164),
 activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller (p. 3198),
 activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller
 (p. 148),
 activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller
 (p. 183),
 activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller
 (p. 286),
 activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller
 (p. 308),
 activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller
 (p. 348),
 activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMessageMarshaller
 (p. 384),
 activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller
 (p. 437),
 activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller
 (p. 482),
 activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller
 (p. 506),
 activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller
 (p. 531),
 activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller
 (p. 554),
 activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller (p. 695),
 activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller (p. 723),
 activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller (p. 1061),
 activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller (p. 1084),
 activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller (p. 1110),
 activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller (p. 1136),
 activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller (p. 1172),
 activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller (p. 1196),
 activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller (p. 1224),
 activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller (p. 1247),
 activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller (p. 1272),
 activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller (p. 1311),
 activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller (p. 1400),
 activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller (p. 1425),
 activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller (p. 1496),

activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller (p. 1585),
 activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller (p. 1679),
 activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller (p. 1731),
 activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller (p. 1756),
 activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller (p. 1779),
 activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller (p. 1798),
 activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller (p. 1825),
 activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller
 (p. 1852), activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller
 (p. 1883), activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller
 (p. 2069), activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller
 (p. 2105), activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller
 (p. 2129), activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller (p. 2155),
 activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller (p. 2212),
 activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller (p. 2255),
 activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller (p. 2327),
 activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller (p. 2429),
 activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller (p. 2455),
 activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller (p. 2479),
 activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller (p. 2553),
 activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller
 (p. 2577), activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller
 (p. 2592), activemq::wireformat::openwire::marshal::v3::ResponseMarshaller (p. 2625),
 activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller (p. 2699),
 activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller (p. 2722),
 activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller (p. 2769),
 activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller (p. 2916),
 activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller (p. 3046),
 activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller (p. 3180),
 activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller (p. 3202),
 activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller
 (p. 156), activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller
 (p. 191), activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller
 (p. 294), activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller
 (p. 316), activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller
 (p. 356), activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller
 (p. 392), activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller
 (p. 445), activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller
 (p. 490), activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller
 (p. 514), activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller
 (p. 539), activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller
 (p. 562), activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller (p. 703),
 activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller (p. 731),
 activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller (p. 1069),
 activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller (p. 1092),
 activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller (p. 1118),
 activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller (p. 1140),
 activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller (p. 1176),
 activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller (p. 1200),
 activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller (p. 1232),
 activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller (p. 1251),
 activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller (p. 1276),
 activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller (p. 1323),
 activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller (p. 1404),

activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller (p. 1429),
 activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller (p. 1500),
 activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller (p. 1589),
 activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller (p. 1683),
 activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller (p. 1727),
 activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller (p. 1760),
 activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller (p. 1775),
 activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller (p. 1802),
 activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller (p. 1829),
 activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller
 (p. 1856),
 activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller
 (p. 1887),
 activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller
 (p. 2073),
 activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller
 (p. 2101),
 activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller
 (p. 2125),
 activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller (p. 2151),
 activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller (p. 2224),
 activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller (p. 2267),
 activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller (p. 2332),
 activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller (p. 2433),
 activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller (p. 2467),
 activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller (p. 2487),
 activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller (p. 2557),
 activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller
 (p. 2581),
 activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller
 (p. 2600),
 activemq::wireformat::openwire::marshal::v4::ResponseMarshaller (p. 2639),
 activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller (p. 2695),
 activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller (p. 2714),
 activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller (p. 2765),
 activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller (p. 2924),
 activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller (p. 3054),
 activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller (p. 3176),
 activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller (p. 3206),
 activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller
 (p. 160),
 activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller
 (p. 195),
 activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller
 (p. 298),
 activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller
 (p. 320),
 activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller
 (p. 360),
 activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller
 (p. 396),
 activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller
 (p. 449),
 activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller
 (p. 494),
 activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller
 (p. 518),
 activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller
 (p. 543),
 activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller
 (p. 566),
 activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller (p. 707),
 activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller (p. 735),
 activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller (p. 1073),
 activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller (p. 1096),
 activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller (p. 1122),
 activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller (p. 1148),
 activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller (p. 1184),
 activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller (p. 1208),
 activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller (p. 1236),
 activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller (p. 1259),
 activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller (p. 1284),

activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller (p. 1319),
 activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller (p. 1412),
 activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller (p. 1433),
 activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller (p. 1504),
 activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller (p. 1593),
 activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller (p. 1687),
 activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller (p. 1739),
 activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller (p. 1764),
 activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller (p. 1787),
 activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller (p. 1810),
 activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller (p. 1821),
 activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller (p. 1860),
 activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller (p. 1895),
 activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller (p. 2065),
 activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller (p. 2113),
 activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller (p. 2137),
 activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller (p. 2159),
 activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller (p. 2220),
 activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller (p. 2259),
 activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller (p. 2340),
 activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller (p. 2437),
 activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller (p. 2459),
 activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller (p. 2483),
 activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller (p. 2545),
 activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller (p. 2565),
 activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller (p. 2596),
 activemq::wireformat::openwire::marshal::v5::ResponseMarshaller (p. 2630),
 activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller (p. 2683),
 activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller (p. 2718),
 activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller (p. 2777),
 activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller (p. 2920),
 activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller (p. 3042),
 activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller (p. 3168), and
 activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller (p. 3214).

6.263.3.2 virtual unsigned char
 activemq::wireformat::openwire::marshal::DataStreamMarshaller::getDataStructureType
 () const [pure virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns

byte Id of this classes DataStructureType

Implemented in activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller
 (p. 152), activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller
 (p. 187), activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller
 (p. 290), activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller
 (p. 312), activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller
 (p. 352), activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller
 (p. 388), activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller
 (p. 441), activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller

(p. 486), activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller
 (p. 510), activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller
 (p. 535), activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller
 (p. 559), activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller (p. 699),
 activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller (p. 727), ac-
 tivemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller (p. 1065),
 activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller (p. 1089),
 activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller (p. 1114),
 activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller (p. 1144),
 activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller (p. 1180),
 activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller (p. 1204),
 activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller (p. 1228), ac-
 tivemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller (p. 1255), ac-
 tivemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller (p. 1280),
 activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller (p. 1315),
 activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller (p. 1408),
 activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller (p. 1437), ac-
 tivemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller (p. 1492),
 activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller (p. 1581),
 activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller (p. 1675),
 activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller (p. 1735),
 activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller (p. 1752),
 activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller (p. 1783), ac-
 tivemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller (p. 1806),
 activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller (p. 1833),
 activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller
 (p. 1848), activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller
 (p. 1891), activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller
 (p. 2077), activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller
 (p. 2109), activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller
 (p. 2133), activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller (p. 2163),
 activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller (p. 2216), ac-
 tivemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller (p. 2263),
 activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller (p. 2336),
 activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller (p. 2441),
 activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller (p. 2463),
 activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller (p. 2491),
 activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller (p. 2542), ac-
 tivemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller
 (p. 2573), activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller
 (p. 2604), activemq::wireformat::openwire::marshal::v1::ResponseMarshaller (p. 2635),
 activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller (p. 2687), ac-
 tivemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller (p. 2710), ac-
 tivemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller (p. 2761), ac-
 tivemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller (p. 2913),
 activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller (p. 3050),
 activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller (p. 3172),
 activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller (p. 3210),
 activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller
 (p. 164), activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller
 (p. 199), activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller
 (p. 302), activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller
 (p. 324), activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller
 (p. 364), activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller

(p. 400), `activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller`
 (p. 453), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller`
 (p. 498), `activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller`
 (p. 522), `activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller`
 (p. 547), `activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller`
 (p. 570), `activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller` (p. 711),
`activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller` (p. 739), `ac-`
`tivemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller` (p. 1077),
`activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller` (p. 1100),
`activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller` (p. 1126),
`activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller` (p. 1152),
`activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller` (p. 1188),
`activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller` (p. 1212),
`activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller` (p. 1240), `ac-`
`tivemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller` (p. 1263), `ac-`
`tivemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller` (p. 1288),
`activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller` (p. 1327),
`activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller` (p. 1397),
`activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller` (p. 1421), `ac-`
`tivemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller` (p. 1488),
`activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller` (p. 1577),
`activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller` (p. 1671),
`activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller` (p. 1723),
`activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller` (p. 1748),
`activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller` (p. 1771), `ac-`
`tivemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller` (p. 1794),
`activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller` (p. 1817),
`activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller`
 (p. 1844), `activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller`
 (p. 1879), `activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller`
 (p. 2062), `activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller`
 (p. 2097), `activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller`
 (p. 2122), `activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller` (p. 2147),
`activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller` (p. 2209), `ac-`
`tivemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller` (p. 2251),
`activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller` (p. 2323),
`activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller` (p. 2425),
`activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller` (p. 2451),
`activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller` (p. 2475),
`activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller` (p. 2549), `ac-`
`tivemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller`
 (p. 2569), `activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller`
 (p. 2589), `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 2621),
`activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller` (p. 2691), `ac-`
`tivemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller` (p. 2706), `ac-`
`tivemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller` (p. 2773), `ac-`
`tivemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller` (p. 2928),
`activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller` (p. 3058),
`activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller` (p. 3164),
`activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller` (p. 3198),
`activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller`
 (p. 148), `activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller`
 (p. 183), `activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller`
 (p. 286), `activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller`

(p. 308), activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller
 (p. 348), activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller
 (p. 384), activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller
 (p. 437), activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller
 (p. 482), activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller
 (p. 506), activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller
 (p. 531), activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller
 (p. 555), activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller (p. 695),
 activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller (p. 723), ac-
 tivemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller (p. 1061),
 activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller (p. 1085),
 activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller (p. 1110),
 activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller (p. 1136),
 activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller (p. 1172),
 activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller (p. 1196),
 activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller (p. 1224), ac-
 tivemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller (p. 1248), ac-
 tivemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller (p. 1272),
 activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller (p. 1311),
 activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller (p. 1401),
 activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller (p. 1425), ac-
 tivemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller (p. 1496),
 activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller (p. 1585),
 activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller (p. 1679),
 activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller (p. 1731),
 activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller (p. 1756),
 activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller (p. 1779), ac-
 tivemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller (p. 1798),
 activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller (p. 1825),
 activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller
 (p. 1852), activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller
 (p. 1883), activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller
 (p. 2069), activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller
 (p. 2105), activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller
 (p. 2129), activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller (p. 2155),
 activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller (p. 2212), ac-
 tivemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller (p. 2255),
 activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller (p. 2328),
 activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller (p. 2429),
 activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller (p. 2455),
 activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller (p. 2479),
 activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller (p. 2553), ac-
 tivemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller
 (p. 2577), activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller
 (p. 2592), activemq::wireformat::openwire::marshal::v3::ResponseMarshaller (p. 2626),
 activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller (p. 2699), ac-
 tivemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller (p. 2722), ac-
 tivemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller (p. 2769), ac-
 tivemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller (p. 2917),
 activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller (p. 3046),
 activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller (p. 3180),
 activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller (p. 3202),
 activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller
 (p. 156), activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller

(p. 191), `activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller`
 (p. 294), `activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller`
 (p. 316), `activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller`
 (p. 356), `activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller`
 (p. 392), `activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller`
 (p. 445), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller`
 (p. 490), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller`
 (p. 514), `activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller`
 (p. 539), `activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller`
 (p. 562), `activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller` (p. 703),
`activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller` (p. 731), `ac-`
`tivemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller` (p. 1069),
`activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller` (p. 1092),
`activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller` (p. 1118),
`activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller` (p. 1140),
`activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller` (p. 1176),
`activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller` (p. 1200),
`activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller` (p. 1232), `ac-`
`tivemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller` (p. 1252), `ac-`
`tivemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller` (p. 1276),
`activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller` (p. 1323),
`activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller` (p. 1404),
`activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller` (p. 1429), `ac-`
`tivemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller` (p. 1500),
`activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller` (p. 1589),
`activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller` (p. 1683),
`activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller` (p. 1727),
`activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller` (p. 1760),
`activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller` (p. 1775), `ac-`
`tivemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller` (p. 1802),
`activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller` (p. 1829),
`activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller`
 (p. 1856), `activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller`
 (p. 1887), `activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller`
 (p. 2073), `activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller`
 (p. 2101), `activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller`
 (p. 2125), `activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller` (p. 2151),
`activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller` (p. 2224), `ac-`
`tivemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller` (p. 2267),
`activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller` (p. 2332),
`activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller` (p. 2433),
`activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller` (p. 2467),
`activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller` (p. 2487),
`activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller` (p. 2557), `ac-`
`tivemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller`
 (p. 2581), `activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller`
 (p. 2600), `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 2639),
`activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller` (p. 2695), `ac-`
`tivemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller` (p. 2714), `ac-`
`tivemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller` (p. 2765), `ac-`
`tivemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller` (p. 2924),
`activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller` (p. 3054),
`activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller` (p. 3176),
`activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller` (p. 3206),

activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller
 (p. 160), activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller
 (p. 195), activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller
 (p. 298), activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller
 (p. 320), activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller
 (p. 360), activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller
 (p. 396), activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller
 (p. 449), activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller
 (p. 494), activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller
 (p. 518), activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller
 (p. 543), activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller
 (p. 566), activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller (p. 707),
 activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller (p. 735), ac-
 tivemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller (p. 1073),
 activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller (p. 1096),
 activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller (p. 1122),
 activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller (p. 1148),
 activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller (p. 1184),
 activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller (p. 1208),
 activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller (p. 1236), ac-
 tivemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller (p. 1259), ac-
 tivemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller (p. 1284),
 activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller (p. 1319),
 activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller (p. 1412),
 activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller (p. 1433), ac-
 tivemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller (p. 1504),
 activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller (p. 1593),
 activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller (p. 1687),
 activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller (p. 1739),
 activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller (p. 1764),
 activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller (p. 1787), ac-
 tivemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller (p. 1810),
 activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller (p. 1821),
 activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller
 (p. 1860), activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller
 (p. 1895), activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller
 (p. 2065), activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller
 (p. 2113), activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller
 (p. 2137), activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller (p. 2159),
 activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller (p. 2220), ac-
 tivemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller (p. 2259),
 activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller (p. 2340),
 activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller (p. 2437),
 activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller (p. 2459),
 activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller (p. 2483),
 activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller (p. 2545), ac-
 tivemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller
 (p. 2566), activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller
 (p. 2596), activemq::wireformat::openwire::marshal::v5::ResponseMarshaller (p. 2630),
 activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller (p. 2683), ac-
 tivemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller (p. 2718), ac-
 tivemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller (p. 2777), ac-
 tivemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller (p. 2920),
 activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller (p. 3042), ac-

tivemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller (p. 3168), and
 activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller (p. 3214).

6.263.3.3 virtual void ac-

```
tivemq::wireformat::openwire::marshal::DataStreamMarshaller::looseMarshal  

( OpenWireFormat * format, commands::DataStructure * command,  

  decaf::io::DataOutputStream * ds ) throw ( decaf::io::IOException )  

[pure virtual]
```

Tight Marshal to the given stream.

Parameters

format - The OpenWireFormat properties

command - the object to Marshal

ds - DataOutputStream to marshal to

Exceptions

IOException if an error occurs.

Implemented in activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller (p. 152), activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller (p. 188), activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller (p. 255), activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller (p. 290), activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller (p. 313), activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller (p. 353), activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller (p. 388), activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller (p. 441), activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller (p. 463), activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller (p. 487), activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller (p. 511), activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller (p. 535), activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller (p. 559), activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller (p. 611), activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller (p. 699), activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller (p. 727), activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller (p. 1065), activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller (p. 1089), activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller (p. 1114), activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller (p. 1144), activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller (p. 1180), activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller (p. 1204), activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller (p. 1228), activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller (p. 1256), activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller (p. 1280), activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller (p. 1316), activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller (p. 1409), activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller (p. 1437), activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller (p. 1492), activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller (p. 1581), activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller (p. 1675), activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller (p. 1735),

activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller (p. 1752),
 activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller (p. 1783), ac-
 tivemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller (p. 1806),
 activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller (p. 1833),
 activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller
 (p. 1848), activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller
 (p. 1892), activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller
 (p. 2078), activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller
 (p. 2109), activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller
 (p. 2134), activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller (p. 2163),
 activemq::wireformat::openwire::marshal::v1::MessageMarshaller (p. 2184), ac-
 tivemq::wireformat::openwire::marshal::v1::MessagePullMarshaller (p. 2217), ac-
 tivemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller (p. 2263),
 activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller (p. 2336),
 activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller (p. 2441),
 activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller (p. 2463),
 activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller (p. 2492),
 activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller (p. 2542), ac-
 tivemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller
 (p. 2574), activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller
 (p. 2605), activemq::wireformat::openwire::marshal::v1::ResponseMarshaller (p. 2635),
 activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller (p. 2687), ac-
 tivemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller (p. 2711), ac-
 tivemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller (p. 2761), ac-
 tivemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller (p. 2913),
 activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller (p. 3023),
 activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller (p. 3050),
 activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller (p. 3172),
 activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller (p. 3211),
 activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller
 (p. 164), activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller
 (p. 200), activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller
 (p. 267), activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller
 (p. 302), activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller
 (p. 325), activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller
 (p. 365), activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller
 (p. 400), activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller
 (p. 453), activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller
 (p. 474), activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller
 (p. 499), activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller
 (p. 523), activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller
 (p. 547), activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller
 (p. 571), activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller
 (p. 631), activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller (p. 711),
 activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller (p. 739), ac-
 tivemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller (p. 1077),
 activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller (p. 1101),
 activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller (p. 1126),
 activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller (p. 1152),
 activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller (p. 1188),
 activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller (p. 1212),
 activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller (p. 1240), ac-
 tivemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller (p. 1264), ac-
 tivemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller (p. 1288),

activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller (p. 1328),
 activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller (p. 1397),
 activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller (p. 1422),
 activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller (p. 1488),
 activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller (p. 1577),
 activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller (p. 1671),
 activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller (p. 1723),
 activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller (p. 1748),
 activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller (p. 1771),
 activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller (p. 1795),
 activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller (p. 1817),
 activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller (p. 1844),
 activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller (p. 1880),
 activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller (p. 2062),
 activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller (p. 2097),
 activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller (p. 2122),
 activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller (p. 2148),
 activemq::wireformat::openwire::marshal::v2::MessageMarshaller (p. 2167),
 activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller (p. 2209),
 activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller (p. 2251),
 activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller (p. 2324),
 activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller (p. 2425),
 activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller (p. 2452),
 activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller (p. 2476),
 activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller (p. 2550),
 activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller (p. 2570),
 activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller (p. 2589),
 activemq::wireformat::openwire::marshal::v2::ResponseMarshaller (p. 2621),
 activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller (p. 2691),
 activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller (p. 2707),
 activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller (p. 2773),
 activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller (p. 2928),
 activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller (p. 3020),
 activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller (p. 3058),
 activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller (p. 3164),
 activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller (p. 3199),
 activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller (p. 148),
 activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller (p. 184),
 activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller (p. 251),
 activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller (p. 286),
 activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller (p. 309),
 activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller (p. 349),
 activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller (p. 384),
 activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller (p. 437),
 activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller (p. 460),
 activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller (p. 483),
 activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller (p. 507),
 activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller (p. 531),
 activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller (p. 555),
 activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller (p. 604),
 activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller (p. 696),
 activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller (p. 723),
 activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller (p. 1061),
 activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller (p. 1085),

activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller (p. 1111),
 activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller (p. 1136),
 activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller (p. 1172),
 activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller (p. 1196),
 activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller (p. 1224),
 activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller (p. 1248),
 activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller (p. 1272),
 activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller (p. 1312),
 activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller (p. 1401),
 activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller (p. 1425),
 activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller (p. 1496),
 activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller (p. 1585),
 activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller (p. 1679),
 activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller (p. 1731),
 activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller (p. 1756),
 activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller (p. 1779),
 activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller (p. 1799),
 activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller (p. 1825),
 activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller
 (p. 1852), activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller
 (p. 1884), activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller
 (p. 2070), activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller
 (p. 2105), activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller
 (p. 2130), activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller (p. 2155),
 activemq::wireformat::openwire::marshal::v3::MessageMarshaller (p. 2176),
 activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller (p. 2213),
 activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller (p. 2255),
 activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller (p. 2328),
 activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller (p. 2429),
 activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller (p. 2455),
 activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller (p. 2480),
 activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller (p. 2554),
 activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller
 (p. 2578), activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller
 (p. 2593), activemq::wireformat::openwire::marshal::v3::ResponseMarshaller (p. 2626),
 activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller (p. 2699),
 activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller (p. 2723),
 activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller (p. 2769),
 activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller (p. 2917),
 activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller (p. 3034),
 activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller (p. 3046),
 activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller (p. 3180),
 activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller (p. 3203),
 activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller
 (p. 156), activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller
 (p. 192), activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller
 (p. 259), activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller
 (p. 294), activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller
 (p. 317), activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller
 (p. 357), activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller
 (p. 392), activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller
 (p. 445), activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller
 (p. 467), activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller
 (p. 491), activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller

(p. 515), `activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller`
 (p. 539), `activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller`
 (p. 563), `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller`
 (p. 618), `activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller` (p. 703),
`activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller` (p. 731), `ac-`
`tivemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller` (p. 1069),
`activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller` (p. 1093),
`activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller` (p. 1118),
`activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller` (p. 1140),
`activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller` (p. 1176),
`activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller` (p. 1200),
`activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller` (p. 1232), `ac-`
`tivemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller` (p. 1252), `ac-`
`tivemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller` (p. 1276),
`activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller` (p. 1324),
`activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller` (p. 1405),
`activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller` (p. 1429), `ac-`
`tivemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller` (p. 1500),
`activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller` (p. 1589),
`activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller` (p. 1683),
`activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller` (p. 1727),
`activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller` (p. 1760),
`activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller` (p. 1775), `ac-`
`tivemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller` (p. 1802),
`activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller` (p. 1829),
`activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller`
 (p. 1856), `activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller`
 (p. 1888), `activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller`
 (p. 2074), `activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller`
 (p. 2101), `activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller`
 (p. 2126), `activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller` (p. 2152),
`activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2172), `ac-`
`tivemq::wireformat::openwire::marshal::v4::MessagePullMarshaller` (p. 2225), `ac-`
`tivemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller` (p. 2267),
`activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller` (p. 2332),
`activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller` (p. 2433),
`activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller` (p. 2467),
`activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller` (p. 2488),
`activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller` (p. 2558), `ac-`
`tivemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller`
 (p. 2582), `activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller`
 (p. 2601), `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 2639),
`activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller` (p. 2695), `ac-`
`tivemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller` (p. 2715), `ac-`
`tivemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller` (p. 2765), `ac-`
`tivemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller` (p. 2924),
`activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller` (p. 3027),
`activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller` (p. 3054),
`activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller` (p. 3176),
`activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller` (p. 3207),
`activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller`
 (p. 160), `activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller`
 (p. 196), `activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller`
 (p. 263), `activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller`

(p. 298), `activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller`
 (p. 321), `activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller`
 (p. 361), `activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller`
 (p. 396), `activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller`
 (p. 449), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller`
 (p. 471), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller`
 (p. 495), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller`
 (p. 519), `activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller`
 (p. 543), `activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller`
 (p. 567), `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller`
 (p. 624), `activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller` (p. 707),
`activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller` (p. 735), `ac-`
`tivemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller` (p. 1073),
`activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller` (p. 1097),
`activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller` (p. 1122),
`activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller` (p. 1148),
`activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller` (p. 1184),
`activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller` (p. 1208),
`activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller` (p. 1236), `ac-`
`tivemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller` (p. 1260), `ac-`
`tivemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller` (p. 1284),
`activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller` (p. 1320),
`activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller` (p. 1413),
`activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller` (p. 1433), `ac-`
`tivemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller` (p. 1504),
`activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller` (p. 1593),
`activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller` (p. 1687),
`activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller` (p. 1739),
`activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller` (p. 1764),
`activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller` (p. 1787), `ac-`
`tivemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller` (p. 1810),
`activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller` (p. 1821),
`activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller`
 (p. 1860), `activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller`
 (p. 1896), `activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller`
 (p. 2066), `activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller`
 (p. 2113), `activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller`
 (p. 2138), `activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller` (p. 2159),
`activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2180), `ac-`
`tivemq::wireformat::openwire::marshal::v5::MessagePullMarshaller` (p. 2221), `ac-`
`tivemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller` (p. 2259),
`activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller` (p. 2341),
`activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller` (p. 2437),
`activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller` (p. 2459),
`activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller` (p. 2484),
`activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller` (p. 2546), `ac-`
`tivemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller`
 (p. 2566), `activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller`
 (p. 2597), `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 2630),
`activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller` (p. 2683), `ac-`
`tivemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller` (p. 2719), `ac-`
`tivemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller` (p. 2777), `ac-`
`tivemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller` (p. 2921),
`activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller` (p. 3031), `ac-`

`tivemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller` (p. 3042), `ativemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller` (p. 3168), and `ativemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller` (p. 3215).

6.263.3.4 `virtual void ativemq::wireformat::openwire::marshal::DataStreamMarshaller::looseUnmarshal (OpenWireFormat * format, commands::DataStructure * command, decaf::io::DataInputStream * dis) throw (decaf::io::IOException)`
[pure virtual]

Loose Un-marhsal to the given stream.

Parameters

format - The OpenWireFormat properties
command - the object to Un-Marshal
dis - the DataInputStream to Un-Marshal from

Exceptions

IOException if an error occurs.

Implemented in `ativemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller` (p. 153), `ativemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller` (p. 188), `ativemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller` (p. 256), `ativemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller` (p. 291), `ativemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller` (p. 313), `ativemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller` (p. 353), `ativemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller` (p. 389), `ativemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller` (p. 442), `ativemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller` (p. 464), `ativemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller` (p. 487), `ativemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller` (p. 511), `ativemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller` (p. 536), `ativemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller` (p. 559), `ativemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 612), `ativemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller` (p. 700), `ativemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller` (p. 728), `ativemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller` (p. 1066), `ativemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller` (p. 1089), `ativemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller` (p. 1115), `ativemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller` (p. 1145), `ativemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller` (p. 1181), `ativemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller` (p. 1205), `ativemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller` (p. 1229), `ativemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller` (p. 1256), `ativemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller` (p. 1281), `ativemq::wireformat::openwire::marshal::v1::DataResponseMarshaller` (p. 1316), `ativemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller` (p. 1409), `ativemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller` (p. 1438), `ativemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller` (p. 1493), `ativemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller` (p. 1582), `ativemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller` (p. 1676),

activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller (p. 1736),
 activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller (p. 1753),
 activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller (p. 1784),
 activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller (p. 1807),
 activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller (p. 1834),
 activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller
 (p. 1849),
 activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller
 (p. 1892),
 activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller
 (p. 2078),
 activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller
 (p. 2110),
 activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller
 (p. 2134),
 activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller (p. 2164),
 activemq::wireformat::openwire::marshal::v1::MessageMarshaller (p. 2184),
 activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller (p. 2217),
 activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller (p. 2264),
 activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller (p. 2337),
 activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller (p. 2442),
 activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller (p. 2464),
 activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller (p. 2492),
 activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller (p. 2542),
 activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller
 (p. 2574),
 activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller
 (p. 2605),
 activemq::wireformat::openwire::marshal::v1::ResponseMarshaller (p. 2635),
 activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller (p. 2688),
 activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller (p. 2711),
 activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller (p. 2762),
 activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller (p. 2913),
 activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller (p. 3024),
 activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller (p. 3051),
 activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller (p. 3173),
 activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller (p. 3211),
 activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller
 (p. 165),
 activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller
 (p. 200),
 activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller
 (p. 267),
 activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller
 (p. 303),
 activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller
 (p. 325),
 activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller
 (p. 365),
 activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller
 (p. 401),
 activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller
 (p. 454),
 activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller
 (p. 475),
 activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller
 (p. 499),
 activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller
 (p. 523),
 activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller
 (p. 548),
 activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller
 (p. 571),
 activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller
 (p. 632),
 activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller (p. 712),
 activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller (p. 740),
 activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller (p. 1078),
 activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller (p. 1101),
 activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller (p. 1127),
 activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller (p. 1153),
 activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller (p. 1189),
 activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller (p. 1213),
 activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller (p. 1241),
 activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller (p. 1264), ac-

tivemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller (p. 1289),
 activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller (p. 1328),
 activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller (p. 1397),
 activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller (p. 1422),
 activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller (p. 1489),
 activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller (p. 1578),
 activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller (p. 1672),
 activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller (p. 1724),
 activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller (p. 1749),
 activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller (p. 1772),
 activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller (p. 1795),
 activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller (p. 1818),
 activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller (p. 1845),
 activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller (p. 1880),
 activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller (p. 2062),
 activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller (p. 2098),
 activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller (p. 2122),
 activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller (p. 2148),
 activemq::wireformat::openwire::marshal::v2::MessageMarshaller (p. 2168),
 activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller (p. 2209),
 activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller (p. 2252),
 activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller (p. 2324),
 activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller (p. 2426),
 activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller (p. 2452),
 activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller (p. 2476),
 activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller (p. 2550),
 activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller (p. 2570),
 activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller (p. 2589),
 activemq::wireformat::openwire::marshal::v2::ResponseMarshaller (p. 2622),
 activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller (p. 2692),
 activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller (p. 2707),
 activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller (p. 2774),
 activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller (p. 2929),
 activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller (p. 3020),
 activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller (p. 3059),
 activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller (p. 3165),
 activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller (p. 3199),
 activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller (p. 149),
 activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller (p. 184),
 activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller (p. 252),
 activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller (p. 287),
 activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller (p. 309),
 activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller (p. 349),
 activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller (p. 385),
 activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller (p. 438),
 activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller (p. 460),
 activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller (p. 483),
 activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller (p. 507),
 activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller (p. 532),
 activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller (p. 555),
 activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller (p. 605),
 activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller (p. 696),
 activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller (p. 724),
 activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller (p. 1062),

activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller (p. 1085),
 activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller (p. 1111),
 activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller (p. 1137),
 activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller (p. 1173),
 activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller (p. 1197),
 activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller (p. 1225),
 activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller (p. 1248),
 activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller (p. 1273),
 activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller (p. 1312),
 activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller (p. 1401),
 activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller (p. 1426),
 activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller (p. 1497),
 activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller (p. 1586),
 activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller (p. 1680),
 activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller (p. 1732),
 activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller (p. 1757),
 activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller (p. 1780),
 activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller (p. 1799),
 activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller (p. 1826),
 activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller (p. 1853),
 activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller (p. 1884),
 activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller (p. 2070),
 activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller (p. 2106),
 activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller (p. 2130),
 activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller (p. 2156),
 activemq::wireformat::openwire::marshal::v3::MessageMarshaller (p. 2176),
 activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller (p. 2213),
 activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller (p. 2256),
 activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller (p. 2328),
 activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller (p. 2430),
 activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller (p. 2456),
 activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller (p. 2480),
 activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller (p. 2554),
 activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller (p. 2578),
 activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller (p. 2593),
 activemq::wireformat::openwire::marshal::v3::ResponseMarshaller (p. 2626),
 activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller (p. 2700),
 activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller (p. 2723),
 activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller (p. 2770),
 activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller (p. 2917),
 activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller (p. 3035),
 activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller (p. 3047),
 activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller (p. 3181),
 activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller (p. 3203),
 activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller (p. 157),
 activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller (p. 192),
 activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller (p. 259),
 activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller (p. 295),
 activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller (p. 317),
 activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller (p. 357),
 activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller (p. 393),
 activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller (p. 446),
 activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller (p. 468),
 activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller

(p. 491), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller`
 (p. 515), `activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller`
 (p. 540), `activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller`
 (p. 563), `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller`
 (p. 619), `activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller` (p. 704),
`activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller` (p. 732), `ac-`
`tivemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller` (p. 1070),
`activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller` (p. 1093),
`activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller` (p. 1119),
`activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller` (p. 1141),
`activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller` (p. 1177),
`activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller` (p. 1201),
`activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller` (p. 1233), `ac-`
`tivemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller` (p. 1252), `ac-`
`tivemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller` (p. 1277),
`activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller` (p. 1324),
`activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller` (p. 1405),
`activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller` (p. 1430), `ac-`
`tivemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller` (p. 1501),
`activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller` (p. 1590),
`activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller` (p. 1684),
`activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller` (p. 1728),
`activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller` (p. 1761),
`activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller` (p. 1776), `ac-`
`tivemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller` (p. 1803),
`activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller` (p. 1830),
`activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller`
 (p. 1857), `activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller`
 (p. 1888), `activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller`
 (p. 2074), `activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller`
 (p. 2102), `activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller`
 (p. 2126), `activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller` (p. 2152),
`activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2172), `ac-`
`tivemq::wireformat::openwire::marshal::v4::MessagePullMarshaller` (p. 2225), `ac-`
`tivemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller` (p. 2268),
`activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller` (p. 2333),
`activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller` (p. 2434),
`activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller` (p. 2468),
`activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller` (p. 2488),
`activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller` (p. 2558), `ac-`
`tivemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller`
 (p. 2582), `activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller`
 (p. 2601), `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 2640),
`activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller` (p. 2696), `ac-`
`tivemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller` (p. 2715), `ac-`
`tivemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller` (p. 2766), `ac-`
`tivemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller` (p. 2925),
`activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller` (p. 3027),
`activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller` (p. 3055),
`activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller` (p. 3177),
`activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller` (p. 3207),
`activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller`
 (p. 161), `activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller`
 (p. 196), `activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller`

(p. 263), `activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller`
 (p. 299), `activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller`
 (p. 321), `activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller`
 (p. 361), `activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller`
 (p. 397), `activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller`
 (p. 450), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller`
 (p. 471), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller`
 (p. 495), `activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller`
 (p. 519), `activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller`
 (p. 544), `activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller`
 (p. 567), `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller`
 (p. 625), `activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller` (p. 708),
`activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller` (p. 736), `ac-`
`tivemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller` (p. 1074),
`activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller` (p. 1097),
`activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller` (p. 1123),
`activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller` (p. 1149),
`activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller` (p. 1185),
`activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller` (p. 1209),
`activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller` (p. 1237), `ac-`
`tivemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller` (p. 1260), `ac-`
`tivemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller` (p. 1285),
`activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller` (p. 1320),
`activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller` (p. 1413),
`activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller` (p. 1434), `ac-`
`tivemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller` (p. 1505),
`activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller` (p. 1594),
`activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller` (p. 1688),
`activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller` (p. 1740),
`activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller` (p. 1765),
`activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller` (p. 1788), `ac-`
`tivemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller` (p. 1811),
`activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller` (p. 1822),
`activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller`
 (p. 1861), `activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller`
 (p. 1896), `activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller`
 (p. 2066), `activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller`
 (p. 2114), `activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller`
 (p. 2138), `activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller` (p. 2160),
`activemq::wireformat::openwire::marshal::v5::MessageMarshaller` (p. 2180), `ac-`
`tivemq::wireformat::openwire::marshal::v5::MessagePullMarshaller` (p. 2221), `ac-`
`tivemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller` (p. 2260),
`activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller` (p. 2341),
`activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller` (p. 2438),
`activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller` (p. 2460),
`activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller` (p. 2484),
`activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller` (p. 2546), `ac-`
`tivemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller`
 (p. 2566), `activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller`
 (p. 2597), `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 2631),
`activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller` (p. 2684), `ac-`
`tivemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller` (p. 2719), `ac-`
`tivemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller` (p. 2778), `ac-`
`tivemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller` (p. 2921),

`activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller` (p. 3031), `activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller` (p. 3043), `activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller` (p. 3169), and `activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller` (p. 3215).

6.263.3.5 `virtual int activemq::wireformat::openwire::marshal::DataStreamMarshaller::tightMarshal1 (OpenWireFormat * format, commands::DataStructure * command, utils::BooleanStream * bs) throw (decaf::io::IOException)` [pure virtual]

Tight Marshal to the given stream.

Parameters

format - The OpenWireFormat properties

command - the object to Marshal

bs - boolean stream to marshal to.

Exceptions

IOException if an error occurs.

Implemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller` (p. 153), `activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller` (p. 188), `activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller` (p. 256), `activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller` (p. 291), `activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller` (p. 313), `activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller` (p. 353), `activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller` (p. 389), `activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller` (p. 442), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller` (p. 464), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller` (p. 487), `activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller` (p. 511), `activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller` (p. 536), `activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller` (p. 560), `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 613), `activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller` (p. 700), `activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller` (p. 728), `activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller` (p. 1066), `activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller` (p. 1090), `activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller` (p. 1115), `activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller` (p. 1145), `activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller` (p. 1181), `activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller` (p. 1205), `activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller` (p. 1229), `activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller` (p. 1256), `activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller` (p. 1281), `activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller` (p. 1316), `activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller` (p. 1409), `activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller` (p. 1438), `activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller` (p. 1493), `activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller` (p. 1582),

activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller (p. 1676),
 activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller (p. 1736),
 activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller (p. 1753),
 activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller (p. 1784),
 activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller (p. 1807),
 activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller (p. 1834),
 activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller
 (p. 1849),
 activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller
 (p. 1892),
 activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller
 (p. 2078),
 activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller
 (p. 2110),
 activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller
 (p. 2134),
 activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller (p. 2164),
 activemq::wireformat::openwire::marshal::v1::MessageMarshaller (p. 2185),
 activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller (p. 2217),
 activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller (p. 2264),
 activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller (p. 2337),
 activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller (p. 2442),
 activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller (p. 2464),
 activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller (p. 2492),
 activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller (p. 2543),
 activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller
 (p. 2574),
 activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller
 (p. 2605),
 activemq::wireformat::openwire::marshal::v1::ResponseMarshaller (p. 2636),
 activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller (p. 2688),
 activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller (p. 2711),
 activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller (p. 2762),
 activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller (p. 2914),
 activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller (p. 3024),
 activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller (p. 3051),
 activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller (p. 3173),
 activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller (p. 3211),
 activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller
 (p. 165),
 activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller
 (p. 200),
 activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller
 (p. 268),
 activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller
 (p. 303),
 activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller
 (p. 325),
 activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller
 (p. 365),
 activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller
 (p. 401),
 activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller
 (p. 454),
 activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller
 (p. 475),
 activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller
 (p. 499),
 activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller
 (p. 523),
 activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller
 (p. 548),
 activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller
 (p. 571),
 activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller
 (p. 633),
 activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller (p. 712),
 activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller (p. 740),
 activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller (p. 1078),
 activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller (p. 1101),
 activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller (p. 1127),
 activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller (p. 1153),
 activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller (p. 1189),
 activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller (p. 1213),
 activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller (p. 1241), ac-

activemq:wireformat::openwire::marshal:v2::ControlCommandMarshaller (p. 1264),
 activemq:wireformat::openwire::marshal:v2::DataArrayResponseMarshaller (p. 1289),
 activemq:wireformat::openwire::marshal:v2::DataResponseMarshaller (p. 1328),
 activemq:wireformat::openwire::marshal:v2::DestinationInfoMarshaller (p. 1398),
 activemq:wireformat::openwire::marshal:v2::DiscoveryEventMarshaller (p. 1422),
 activemq:wireformat::openwire::marshal:v2::ExceptionResponseMarshaller (p. 1489),
 activemq:wireformat::openwire::marshal:v2::FlushCommandMarshaller (p. 1578),
 activemq:wireformat::openwire::marshal:v2::IntegerResponseMarshaller (p. 1672),
 activemq:wireformat::openwire::marshal:v2::JournalQueueAckMarshaller (p. 1724),
 activemq:wireformat::openwire::marshal:v2::JournalTopicAckMarshaller (p. 1749),
 activemq:wireformat::openwire::marshal:v2::JournalTraceMarshaller (p. 1772),
 activemq:wireformat::openwire::marshal:v2::JournalTransactionMarshaller (p. 1795),
 activemq:wireformat::openwire::marshal:v2::KeepAliveInfoMarshaller (p. 1818),
 activemq:wireformat::openwire::marshal:v2::LastPartialCommandMarshaller (p. 1845),
 activemq:wireformat::openwire::marshal:v2::LocalTransactionIdMarshaller (p. 1880),
 activemq:wireformat::openwire::marshal:v2::MessageAckMarshaller (p. 2063),
 activemq:wireformat::openwire::marshal:v2::MessageDispatchMarshaller (p. 2098),
 activemq:wireformat::openwire::marshal:v2::MessageDispatchNotificationMarshaller (p. 2123),
 activemq:wireformat::openwire::marshal:v2::MessageIdMarshaller (p. 2148),
 activemq:wireformat::openwire::marshal:v2::MessageMarshaller (p. 2169),
 activemq:wireformat::openwire::marshal:v2::MessagePullMarshaller (p. 2210),
 activemq:wireformat::openwire::marshal:v2::NetworkBridgeFilterMarshaller (p. 2252),
 activemq:wireformat::openwire::marshal:v2::PartialCommandMarshaller (p. 2325),
 activemq:wireformat::openwire::marshal:v2::ProducerAckMarshaller (p. 2426),
 activemq:wireformat::openwire::marshal:v2::ProducerIdMarshaller (p. 2452),
 activemq:wireformat::openwire::marshal:v2::ProducerInfoMarshaller (p. 2476),
 activemq:wireformat::openwire::marshal:v2::RemoveInfoMarshaller (p. 2550),
 activemq:wireformat::openwire::marshal:v2::RemoveSubscriptionInfoMarshaller (p. 2570),
 activemq:wireformat::openwire::marshal:v2::ReplayCommandMarshaller (p. 2590),
 activemq:wireformat::openwire::marshal:v2::ResponseMarshaller (p. 2622),
 activemq:wireformat::openwire::marshal:v2::SessionIdMarshaller (p. 2692),
 activemq:wireformat::openwire::marshal:v2::SessionInfoMarshaller (p. 2707),
 activemq:wireformat::openwire::marshal:v2::ShutdownInfoMarshaller (p. 2774),
 activemq:wireformat::openwire::marshal:v2::SubscriptionInfoMarshaller (p. 2929),
 activemq:wireformat::openwire::marshal:v2::TransactionIdMarshaller (p. 3021),
 activemq:wireformat::openwire::marshal:v2::TransactionInfoMarshaller (p. 3059),
 activemq:wireformat::openwire::marshal:v2::WireFormatInfoMarshaller (p. 3165),
 activemq:wireformat::openwire::marshal:v2::XATransactionIdMarshaller (p. 3199),
 activemq:wireformat::openwire::marshal:v3::ActiveMQBlobMessageMarshaller (p. 149),
 activemq:wireformat::openwire::marshal:v3::ActiveMQBytesMessageMarshaller (p. 184),
 activemq:wireformat::openwire::marshal:v3::ActiveMQDestinationMarshaller (p. 252),
 activemq:wireformat::openwire::marshal:v3::ActiveMQMapMessageMarshaller (p. 287),
 activemq:wireformat::openwire::marshal:v3::ActiveMQMessageMarshaller (p. 309),
 activemq:wireformat::openwire::marshal:v3::ActiveMQObjectMessageMarshaller (p. 349),
 activemq:wireformat::openwire::marshal:v3::ActiveMQQueueMarshaller (p. 385),
 activemq:wireformat::openwire::marshal:v3::ActiveMQStreamMessageMarshaller (p. 438),
 activemq:wireformat::openwire::marshal:v3::ActiveMQTempDestinationMarshaller (p. 461),
 activemq:wireformat::openwire::marshal:v3::ActiveMQTempQueueMarshaller (p. 483),
 activemq:wireformat::openwire::marshal:v3::ActiveMQTempTopicMarshaller (p. 507),
 activemq:wireformat::openwire::marshal:v3::ActiveMQTextMessageMarshaller (p. 532),
 activemq:wireformat::openwire::marshal:v3::ActiveMQTopicMarshaller (p. 556),
 activemq:wireformat::openwire::marshal:v3::BaseCommandMarshaller (p. 606),
 activemq:wireformat::openwire::marshal:v3::BrokerIdMarshaller (p. 696),
 activemq:wireformat::openwire::marshal:v3::BrokerInfoMarshaller (p. 724), ac-

tivemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller (p. 1062),
 activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller (p. 1086),
 activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller (p. 1111),
 activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller (p. 1137),
 activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller (p. 1173),
 activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller (p. 1197),
 activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller (p. 1225),
 activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller (p. 1249),
 activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller (p. 1273),
 activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller (p. 1312),
 activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller (p. 1402),
 activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller (p. 1426),
 activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller (p. 1497),
 activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller (p. 1586),
 activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller (p. 1680),
 activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller (p. 1732),
 activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller (p. 1757),
 activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller (p. 1780),
 activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller (p. 1799),
 activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller (p. 1826),
 activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller (p. 1853),
 activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller (p. 1884),
 activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller (p. 2070),
 activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller (p. 2106),
 activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller (p. 2130),
 activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller (p. 2156),
 activemq::wireformat::openwire::marshal::v3::MessageMarshaller (p. 2177),
 activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller (p. 2213),
 activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller (p. 2256),
 activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller (p. 2329),
 activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller (p. 2430),
 activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller (p. 2456),
 activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller (p. 2480),
 activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller (p. 2554),
 activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller (p. 2578),
 activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller (p. 2593),
 activemq::wireformat::openwire::marshal::v3::ResponseMarshaller (p. 2627),
 activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller (p. 2700),
 activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller (p. 2723),
 activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller (p. 2770),
 activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller (p. 2918),
 activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller (p. 3035),
 activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller (p. 3047),
 activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller (p. 3181),
 activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller (p. 3203),
 activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller (p. 157),
 activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller (p. 192),
 activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller (p. 260),
 activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller (p. 295),
 activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller (p. 317),
 activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller (p. 357),
 activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller (p. 393),
 activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller (p. 446),
 activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller

(p. 468), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller`
 (p. 491), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller`
 (p. 515), `activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller`
 (p. 540), `activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller`
 (p. 563), `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller`
 (p. 620), `activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller` (p. 704),
`activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller` (p. 732), `ac-`
`tivemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller` (p. 1070),
`activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller` (p. 1093),
`activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller` (p. 1119),
`activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller` (p. 1141),
`activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller` (p. 1177),
`activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller` (p. 1201),
`activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller` (p. 1233), `ac-`
`tivemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller` (p. 1253), `ac-`
`tivemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller` (p. 1277),
`activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller` (p. 1324),
`activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller` (p. 1405),
`activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller` (p. 1430), `ac-`
`tivemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller` (p. 1501),
`activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller` (p. 1590),
`activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller` (p. 1684),
`activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller` (p. 1728),
`activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller` (p. 1761),
`activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller` (p. 1776), `ac-`
`tivemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller` (p. 1803),
`activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller` (p. 1830),
`activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller`
 (p. 1857), `activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller`
 (p. 1888), `activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller`
 (p. 2074), `activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller`
 (p. 2102), `activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller`
 (p. 2126), `activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller` (p. 2152),
`activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2173), `ac-`
`tivemq::wireformat::openwire::marshal::v4::MessagePullMarshaller` (p. 2225), `ac-`
`tivemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller` (p. 2268),
`activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller` (p. 2333),
`activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller` (p. 2434),
`activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller` (p. 2468),
`activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller` (p. 2488),
`activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller` (p. 2558), `ac-`
`tivemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller`
 (p. 2582), `activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller`
 (p. 2601), `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 2640),
`activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller` (p. 2696), `ac-`
`tivemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller` (p. 2715), `ac-`
`tivemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller` (p. 2766), `ac-`
`tivemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller` (p. 2925),
`activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller` (p. 3028),
`activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller` (p. 3055),
`activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller` (p. 3177),
`activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller` (p. 3207),
`activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller`
 (p. 161), `activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller`

(p. 196), activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller
 (p. 264), activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller
 (p. 299), activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller
 (p. 321), activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller
 (p. 361), activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller
 (p. 397), activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller
 (p. 450), activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller
 (p. 472), activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller
 (p. 495), activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller
 (p. 519), activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller
 (p. 544), activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller
 (p. 567), activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller
 (p. 626), activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller (p. 708),
 activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller (p. 736), ac-
 tivemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller (p. 1074),
 activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller (p. 1097),
 activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller (p. 1123),
 activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller (p. 1149),
 activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller (p. 1185),
 activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller (p. 1209),
 activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller (p. 1237), ac-
 tivemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller (p. 1260), ac-
 tivemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller (p. 1285),
 activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller (p. 1320),
 activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller (p. 1413),
 activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller (p. 1434), ac-
 tivemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller (p. 1505),
 activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller (p. 1594),
 activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller (p. 1688),
 activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller (p. 1740),
 activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller (p. 1765),
 activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller (p. 1788), ac-
 tivemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller (p. 1811),
 activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller (p. 1822),
 activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller
 (p. 1861), activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller
 (p. 1896), activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller
 (p. 2066), activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller
 (p. 2114), activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller
 (p. 2138), activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller (p. 2160),
 activemq::wireformat::openwire::marshal::v5::MessageMarshaller (p. 2181), ac-
 tivemq::wireformat::openwire::marshal::v5::MessagePullMarshaller (p. 2221), ac-
 tivemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller (p. 2260),
 activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller (p. 2341),
 activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller (p. 2438),
 activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller (p. 2460),
 activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller (p. 2484),
 activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller (p. 2546), ac-
 tivemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller
 (p. 2567), activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller
 (p. 2597), activemq::wireformat::openwire::marshal::v5::ResponseMarshaller (p. 2631),
 activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller (p. 2684), ac-
 tivemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller (p. 2719), ac-
 tivemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller (p. 2778), ac-

tivemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller (p. 2921),
 activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller (p. 3032), ac-
 tivemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller (p. 3043), ac-
 tivemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller (p. 3169), and
 activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller (p. 3215).

6.263.3.6 virtual void ac-

```
tivemq::wireformat::openwire::marshal::DataStreamMarshaller::tightMarshal2
( OpenWireFormat * format, commands::DataStructure * command,
  decaf::io::DataOutputStream * ds, utils::BooleanStream * bs ) throw (
  decaf::io::IOException ) [pure virtual]
```

Tight Marshal to the given stream.

Parameters

format - The OpenWireFormat properties

command - the object to Marshal

ds - the DataOutputStream to Marshal to

bs - boolean stream to marshal to.

Exceptions

IOException if an error occurs.

Implemented in activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller
 (p. 153), activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller
 (p. 189), activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller
 (p. 257), activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller
 (p. 291), activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller
 (p. 314), activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller
 (p. 354), activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller
 (p. 389), activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller
 (p. 442), activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller
 (p. 465), activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller
 (p. 488), activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller
 (p. 512), activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller
 (p. 536), activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller
 (p. 560), activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller
 (p. 614), activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller (p. 701),
 activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller (p. 728), ac-
 tivemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller (p. 1066),
 activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller (p. 1090),
 activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller (p. 1116),
 activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller (p. 1145),
 activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller (p. 1181),
 activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller (p. 1205),
 activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller (p. 1229), ac-
 tivemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller (p. 1257), ac-
 tivemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller (p. 1282),
 activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller (p. 1317),
 activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller (p. 1410),

activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller (p. 1438),
 activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller (p. 1494),
 activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller (p. 1582),
 activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller (p. 1677),
 activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller (p. 1736),
 activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller (p. 1753),
 activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller (p. 1784),
 activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller (p. 1807),
 activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller (p. 1834),
 activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller
 (p. 1850), activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller
 (p. 1893), activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller
 (p. 2079), activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller
 (p. 2110), activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller
 (p. 2135), activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller (p. 2164),
 activemq::wireformat::openwire::marshal::v1::MessageMarshaller (p. 2186),
 activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller (p. 2218),
 activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller (p. 2264),
 activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller (p. 2338),
 activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller (p. 2442),
 activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller (p. 2464),
 activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller (p. 2493),
 activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller (p. 2543),
 activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller
 (p. 2575), activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller
 (p. 2606), activemq::wireformat::openwire::marshal::v1::ResponseMarshaller (p. 2636),
 activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller (p. 2688),
 activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller (p. 2712),
 activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller (p. 2762),
 activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller (p. 2914),
 activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller (p. 3025),
 activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller (p. 3051),
 activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller (p. 3173),
 activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller (p. 3212),
 activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller
 (p. 165), activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller
 (p. 201), activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller
 (p. 268), activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller
 (p. 303), activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller
 (p. 326), activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller
 (p. 366), activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller
 (p. 401), activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller
 (p. 454), activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller
 (p. 476), activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller
 (p. 500), activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller
 (p. 524), activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller
 (p. 548), activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller
 (p. 572), activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller
 (p. 634), activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller (p. 712),
 activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller (p. 740),
 activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller (p. 1078),
 activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller (p. 1102),
 activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller (p. 1127),
 activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller (p. 1153),

activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller (p. 1189),
 activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller (p. 1213),
 activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller (p. 1241),
 activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller (p. 1265),
 activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller (p. 1290),
 activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller (p. 1329),
 activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller (p. 1398),
 activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller (p. 1423),
 activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller (p. 1490),
 activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller (p. 1579),
 activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller (p. 1673),
 activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller (p. 1725),
 activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller (p. 1749),
 activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller (p. 1772),
 activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller (p. 1796),
 activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller (p. 1818),
 activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller
 (p. 1846), activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller
 (p. 1881), activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller
 (p. 2063), activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller
 (p. 2098), activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller
 (p. 2123), activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller (p. 2149),
 activemq::wireformat::openwire::marshal::v2::MessageMarshaller (p. 2169),
 activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller (p. 2210),
 activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller (p. 2253),
 activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller (p. 2325),
 activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller (p. 2426),
 activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller (p. 2453),
 activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller (p. 2477),
 activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller (p. 2551),
 activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller
 (p. 2571), activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller
 (p. 2590), activemq::wireformat::openwire::marshal::v2::ResponseMarshaller (p. 2623),
 activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller (p. 2692),
 activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller (p. 2708),
 activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller (p. 2774),
 activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller (p. 2929),
 activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller (p. 3021),
 activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller (p. 3059),
 activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller (p. 3166),
 activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller (p. 3200),
 activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller
 (p. 149), activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller
 (p. 185), activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller
 (p. 253), activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller
 (p. 287), activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller
 (p. 310), activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller
 (p. 350), activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller
 (p. 385), activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller
 (p. 439), activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller
 (p. 461), activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller
 (p. 484), activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller
 (p. 508), activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller
 (p. 532), activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller

(p. 556), activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller
 (p. 608), activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller (p. 697),
 activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller (p. 724), ac-
 tivemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller (p. 1062),
 activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller (p. 1086),
 activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller (p. 1112),
 activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller (p. 1137),
 activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller (p. 1173),
 activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller (p. 1198),
 activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller (p. 1226), ac-
 tivemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller (p. 1249), ac-
 tivemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller (p. 1274),
 activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller (p. 1313),
 activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller (p. 1402),
 activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller (p. 1427), ac-
 tivemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller (p. 1498),
 activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller (p. 1586),
 activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller (p. 1681),
 activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller (p. 1732),
 activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller (p. 1757),
 activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller (p. 1780), ac-
 tivemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller (p. 1800),
 activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller (p. 1826),
 activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller
 (p. 1854), activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller
 (p. 1885), activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller
 (p. 2071), activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller
 (p. 2106), activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller
 (p. 2131), activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller (p. 2157),
 activemq::wireformat::openwire::marshal::v3::MessageMarshaller (p. 2178), ac-
 tivemq::wireformat::openwire::marshal::v3::MessagePullMarshaller (p. 2214), ac-
 tivemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller (p. 2256),
 activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller (p. 2329),
 activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller (p. 2430),
 activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller (p. 2457),
 activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller (p. 2481),
 activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller (p. 2555), ac-
 tivemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller
 (p. 2579), activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller
 (p. 2594), activemq::wireformat::openwire::marshal::v3::ResponseMarshaller (p. 2627),
 activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller (p. 2700), ac-
 tivemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller (p. 2724), ac-
 tivemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller (p. 2770), ac-
 tivemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller (p. 2918),
 activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller (p. 3036),
 activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller (p. 3047),
 activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller (p. 3181),
 activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller (p. 3204),
 activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller
 (p. 157), activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller
 (p. 193), activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller
 (p. 260), activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller
 (p. 295), activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller
 (p. 318), activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller

(p. 358), `activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller`
 (p. 393), `activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller`
 (p. 446), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller`
 (p. 469), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller`
 (p. 492), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller`
 (p. 516), `activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller`
 (p. 540), `activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller`
 (p. 564), `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller`
 (p. 621), `activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller` (p. 704),
`activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller` (p. 732), `ac-`
`tivemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller` (p. 1070),
`activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller` (p. 1094),
`activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller` (p. 1119),
`activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller` (p. 1141),
`activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller` (p. 1177),
`activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller` (p. 1201),
`activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller` (p. 1233), `ac-`
`tivemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller` (p. 1253), `ac-`
`tivemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller` (p. 1278),
`activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller` (p. 1325),
`activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller` (p. 1406),
`activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller` (p. 1430), `ac-`
`tivemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller` (p. 1502),
`activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller` (p. 1590),
`activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller` (p. 1685),
`activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller` (p. 1728),
`activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller` (p. 1761),
`activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller` (p. 1776), `ac-`
`tivemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller` (p. 1804),
`activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller` (p. 1830),
`activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller`
 (p. 1858), `activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller`
 (p. 1889), `activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller`
 (p. 2075), `activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller`
 (p. 2102), `activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller`
 (p. 2127), `activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller` (p. 2153),
`activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2174), `ac-`
`tivemq::wireformat::openwire::marshal::v4::MessagePullMarshaller` (p. 2226), `ac-`
`tivemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller` (p. 2268),
`activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller` (p. 2334),
`activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller` (p. 2434),
`activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller` (p. 2468),
`activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller` (p. 2489),
`activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller` (p. 2559), `ac-`
`tivemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller`
 (p. 2583), `activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller`
 (p. 2602), `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 2641),
`activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller` (p. 2696), `ac-`
`tivemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller` (p. 2716), `ac-`
`tivemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller` (p. 2766), `ac-`
`tivemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller` (p. 2926),
`activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller` (p. 3028),
`activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller` (p. 3055),
`activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller` (p. 3177),

activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller (p. 3208),
 activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller
 (p. 161), activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller
 (p. 197), activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller
 (p. 264), activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller
 (p. 299), activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller
 (p. 322), activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller
 (p. 362), activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller
 (p. 397), activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller
 (p. 450), activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller
 (p. 472), activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller
 (p. 496), activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller
 (p. 520), activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller
 (p. 544), activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller
 (p. 568), activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller
 (p. 628), activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller (p. 708),
 activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller (p. 736), ac-
 tivemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller (p. 1074),
 activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller (p. 1098),
 activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller (p. 1123),
 activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller (p. 1149),
 activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller (p. 1185),
 activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller (p. 1209),
 activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller (p. 1237), ac-
 tivemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller (p. 1261), ac-
 tivemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller (p. 1286),
 activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller (p. 1321),
 activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller (p. 1414),
 activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller (p. 1434), ac-
 tivemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller (p. 1506),
 activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller (p. 1594),
 activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller (p. 1689),
 activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller (p. 1740),
 activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller (p. 1765),
 activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller (p. 1788), ac-
 tivemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller (p. 1811),
 activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller (p. 1822),
 activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller
 (p. 1862), activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller
 (p. 1897), activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller
 (p. 2067), activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller
 (p. 2114), activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller
 (p. 2139), activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller (p. 2160),
 activemq::wireformat::openwire::marshal::v5::MessageMarshaller (p. 2182), ac-
 tivemq::wireformat::openwire::marshal::v5::MessagePullMarshaller (p. 2222), ac-
 tivemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller (p. 2260),
 activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller (p. 2342),
 activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller (p. 2438),
 activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller (p. 2460),
 activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller (p. 2485),
 activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller (p. 2547), ac-
 tivemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller
 (p. 2567), activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller
 (p. 2598), activemq::wireformat::openwire::marshal::v5::ResponseMarshaller (p. 2632),

activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller (p. 2684), activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller (p. 2720), activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller (p. 2778), activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller (p. 2922), activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller (p. 3032), activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller (p. 3044), activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller (p. 3169), and activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller (p. 3216).

6.263.3.7 virtual void activemq::wireformat::openwire::marshal::DataStreamMarshaller::tightUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [pure virtual]

Tight Un-marhsal to the given stream.

Parameters

format - The OpenWireFormat properties
command - the object to Un-Marshal
dis - the DataInputStream to Un-Marshal from
bs - boolean stream to unmarshal from.

Exceptions

IOException if an error occurs.

Implemented in activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller (p. 154), activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller (p. 189), activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller (p. 257), activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller (p. 292), activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller (p. 314), activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller (p. 354), activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller (p. 390), activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller (p. 443), activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller (p. 465), activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller (p. 488), activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller (p. 512), activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller (p. 537), activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller (p. 560), activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller (p. 615), activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller (p. 701), activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller (p. 729), activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller (p. 1067), activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller (p. 1090), activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller (p. 1116), activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller (p. 1146), activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller (p. 1182), activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller (p. 1206), activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller (p. 1230), activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller (p. 1257), activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller (p. 1282),

activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller (p. 1317),
 activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller (p. 1410),
 activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller (p. 1439),
 activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller (p. 1494),
 activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller (p. 1583),
 activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller (p. 1677),
 activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller (p. 1737),
 activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller (p. 1754),
 activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller (p. 1785),
 activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller (p. 1808),
 activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller (p. 1835),
 activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller
 (p. 1850),
 activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller
 (p. 1893),
 activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller
 (p. 2079),
 activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller
 (p. 2111),
 activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller
 (p. 2135),
 activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller (p. 2165),
 activemq::wireformat::openwire::marshal::v1::MessageMarshaller (p. 2186),
 activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller (p. 2218),
 activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller (p. 2265),
 activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller (p. 2338),
 activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller (p. 2443),
 activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller (p. 2465),
 activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller (p. 2493),
 activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller (p. 2543),
 activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller
 (p. 2575),
 activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller
 (p. 2606),
 activemq::wireformat::openwire::marshal::v1::ResponseMarshaller (p. 2637),
 activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller (p. 2689),
 activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller (p. 2712),
 activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller (p. 2763),
 activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller (p. 2915),
 activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller (p. 3025),
 activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller (p. 3052),
 activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller (p. 3174),
 activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller (p. 3212),
 activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller
 (p. 166),
 activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller
 (p. 201),
 activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller
 (p. 269),
 activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller
 (p. 304),
 activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller
 (p. 326),
 activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller
 (p. 366),
 activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller
 (p. 402),
 activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller
 (p. 455),
 activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller
 (p. 476),
 activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller
 (p. 500),
 activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller
 (p. 524),
 activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller
 (p. 549),
 activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller
 (p. 572),
 activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller
 (p. 635),
 activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller (p. 713),
 activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller (p. 741),
 activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller (p. 1079),
 activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller (p. 1102),

activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller (p. 1128),
 activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller (p. 1154),
 activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller (p. 1190),
 activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller (p. 1214),
 activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller (p. 1242),
 activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller (p. 1265),
 activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller (p. 1290),
 activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller (p. 1329),
 activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller (p. 1399),
 activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller (p. 1423),
 activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller (p. 1490),
 activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller (p. 1579),
 activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller (p. 1673),
 activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller (p. 1725),
 activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller (p. 1750),
 activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller (p. 1773),
 activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller (p. 1796),
 activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller (p. 1819),
 activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller (p. 1846),
 activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller (p. 1881),
 activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller (p. 2063),
 activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller (p. 2099),
 activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller (p. 2123),
 activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller (p. 2149),
 activemq::wireformat::openwire::marshal::v2::MessageMarshaller (p. 2170),
 activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller (p. 2210),
 activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller (p. 2253),
 activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller (p. 2326),
 activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller (p. 2427),
 activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller (p. 2453),
 activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller (p. 2477),
 activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller (p. 2551),
 activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller (p. 2571),
 activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller (p. 2590),
 activemq::wireformat::openwire::marshal::v2::ResponseMarshaller (p. 2623),
 activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller (p. 2693),
 activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller (p. 2708),
 activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller (p. 2775),
 activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller (p. 2930),
 activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller (p. 3022),
 activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller (p. 3060),
 activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller (p. 3166),
 activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller (p. 3200),
 activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller (p. 150),
 activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller (p. 185),
 activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller (p. 253),
 activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller (p. 288),
 activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller (p. 310),
 activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller (p. 350),
 activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller (p. 386),
 activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller (p. 439),
 activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller (p. 462),
 activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller (p. 484),
 activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller

(p. 508), activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller
 (p. 533), activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller
 (p. 557), activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller
 (p. 609), activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller (p. 697),
 activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller (p. 725), ac-
 tivemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller (p. 1063),
 activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller (p. 1087),
 activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller (p. 1112),
 activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller (p. 1138),
 activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller (p. 1174),
 activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller (p. 1198),
 activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller (p. 1226), ac-
 tivemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller (p. 1250), ac-
 tivemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller (p. 1274),
 activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller (p. 1313),
 activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller (p. 1402),
 activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller (p. 1427), ac-
 tivemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller (p. 1498),
 activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller (p. 1587),
 activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller (p. 1681),
 activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller (p. 1733),
 activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller (p. 1758),
 activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller (p. 1781), ac-
 tivemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller (p. 1800),
 activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller (p. 1827),
 activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller
 (p. 1854), activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller
 (p. 1885), activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller
 (p. 2071), activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller
 (p. 2107), activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller
 (p. 2131), activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller (p. 2157),
 activemq::wireformat::openwire::marshal::v3::MessageMarshaller (p. 2178), ac-
 tivemq::wireformat::openwire::marshal::v3::MessagePullMarshaller (p. 2214), ac-
 tivemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller (p. 2257),
 activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller (p. 2330),
 activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller (p. 2431),
 activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller (p. 2457),
 activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller (p. 2481),
 activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller (p. 2555), ac-
 tivemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller
 (p. 2579), activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller
 (p. 2594), activemq::wireformat::openwire::marshal::v3::ResponseMarshaller (p. 2628),
 activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller (p. 2701), ac-
 tivemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller (p. 2724), ac-
 tivemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller (p. 2771), ac-
 tivemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller (p. 2918),
 activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller (p. 3036),
 activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller (p. 3048),
 activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller (p. 3182),
 activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller (p. 3204),
 activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller
 (p. 158), activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller
 (p. 193), activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller
 (p. 261), activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller

(p. 296), `activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller`
 (p. 318), `activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller`
 (p. 358), `activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller`
 (p. 394), `activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller`
 (p. 447), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller`
 (p. 469), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller`
 (p. 492), `activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller`
 (p. 516), `activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller`
 (p. 541), `activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller`
 (p. 564), `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller`
 (p. 622), `activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller` (p. 705),
`activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller` (p. 733), `ac-`
`tivemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller` (p. 1071),
`activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller` (p. 1094),
`activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller` (p. 1120),
`activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller` (p. 1142),
`activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller` (p. 1178),
`activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller` (p. 1202),
`activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller` (p. 1234), `ac-`
`tivemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller` (p. 1253), `ac-`
`tivemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller` (p. 1278),
`activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller` (p. 1325),
`activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller` (p. 1406),
`activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller` (p. 1431), `ac-`
`tivemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller` (p. 1502),
`activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller` (p. 1591),
`activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller` (p. 1685),
`activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller` (p. 1729),
`activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller` (p. 1762),
`activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller` (p. 1777), `ac-`
`tivemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller` (p. 1804),
`activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller` (p. 1831),
`activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller`
 (p. 1858), `activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller`
 (p. 1889), `activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller`
 (p. 2075), `activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller`
 (p. 2103), `activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller`
 (p. 2127), `activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller` (p. 2153),
`activemq::wireformat::openwire::marshal::v4::MessageMarshaller` (p. 2174), `ac-`
`tivemq::wireformat::openwire::marshal::v4::MessagePullMarshaller` (p. 2226), `ac-`
`tivemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller` (p. 2269),
`activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller` (p. 2334),
`activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller` (p. 2435),
`activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller` (p. 2469),
`activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller` (p. 2489),
`activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller` (p. 2559), `ac-`
`tivemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller`
 (p. 2583), `activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller`
 (p. 2602), `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 2641),
`activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller` (p. 2697), `ac-`
`tivemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller` (p. 2716), `ac-`
`tivemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller` (p. 2767), `ac-`
`tivemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller` (p. 2926),
`activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller` (p. 3029),

activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller (p. 3056),
 activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller (p. 3178),
 activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller (p. 3208),
 activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller
 (p. 162), activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller
 (p. 197), activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller
 (p. 265), activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller
 (p. 300), activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller
 (p. 322), activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller
 (p. 362), activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller
 (p. 398), activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller
 (p. 451), activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller
 (p. 473), activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller
 (p. 496), activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller
 (p. 520), activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller
 (p. 545), activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller
 (p. 568), activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller
 (p. 629), activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller (p. 709),
 activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller (p. 737), ac-
 tivemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller (p. 1075),
 activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller (p. 1098),
 activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller (p. 1124),
 activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller (p. 1150),
 activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller (p. 1186),
 activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller (p. 1210),
 activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller (p. 1238), ac-
 tivemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller (p. 1261), ac-
 tivemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller (p. 1286),
 activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller (p. 1321),
 activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller (p. 1414),
 activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller (p. 1435), ac-
 tivemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller (p. 1506),
 activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller (p. 1595),
 activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller (p. 1689),
 activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller (p. 1741),
 activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller (p. 1766),
 activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller (p. 1789), ac-
 tivemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller (p. 1812),
 activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller (p. 1823),
 activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller
 (p. 1862), activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller
 (p. 1897), activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller
 (p. 2067), activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller
 (p. 2115), activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller
 (p. 2139), activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller (p. 2161),
 activemq::wireformat::openwire::marshal::v5::MessageMarshaller (p. 2182), ac-
 tivemq::wireformat::openwire::marshal::v5::MessagePullMarshaller (p. 2222), ac-
 tivemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller (p. 2261),
 activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller (p. 2342),
 activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller (p. 2439),
 activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller (p. 2461),
 activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller (p. 2485),
 activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller (p. 2547), ac-
 tivemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller

(p. 2567), `activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller` (p. 2598), `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 2632), `activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller` (p. 2685), `activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller` (p. 2720), `activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller` (p. 2779), `activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller` (p. 2922), `activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller` (p. 3033), `activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller` (p. 3044), `activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller` (p. 3170), and `activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller` (p. 3216).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/DataStreamMarshaller.h`

6.264 `activemq::commands::DataStructure` Class Reference

```
#include <src/main/activemq/commands/DataStructure.h>
```

Inheritance diagram for `activemq::commands::DataStructure`:

Public Member Functions

- virtual `~DataStructure ()`
- virtual unsigned char `getDataStructureType ()` const =0

*Get the **DataStructure** (p. 1372) Type as defined in *CommandTypes.h*.*

- virtual `DataStructure * cloneDataStructure ()` const =0

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void `copyDataStructure (const DataStructure *src)`=0

Copy the contents of the passed object into this objects members, overwriting any existing data.

- virtual std::string `toString ()` const =0

*Returns a string containing the information for this **DataStructure** (p. 1372) such as its type and value of its elements.*

- virtual bool `equals (const DataStructure *value)` const =0

*Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent.*

6.264.1 Constructor & Destructor Documentation

6.264.1.1 virtual `activemq::commands::DataStructure::~~DataStructure ()`
`[inline, virtual]`

6.264.2 Member Function Documentation

6.264.2.1 virtual `DataStructure* activemq::commands::DataStructure::cloneDataStructure ()`
`const [pure virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implemented in `activemq::commands::ActiveMQBlobMessage` (p. 143), `activemq::commands::ActiveMQBytesMessage` (p. 170), `activemq::commands::ActiveMQDestination` (p. 242), `activemq::commands::ActiveMQMapMessage` (p. 276), `activemq::commands::ActiveMQMessage` (p. 306), `activemq::commands::ActiveMQObjectMessage` (p. 345), `activemq::commands::ActiveMQStreamMessage` (p. 425), `activemq::commands::ActiveMQTextMessage` (p. 527), `activemq::commands::BooleanExpression` (p. 679), `activemq::commands::BrokerError` (p. 687), `activemq::commands::BrokerInfo` (p. 715), `activemq::commands::ConnectionControl` (p. 1057), `activemq::commands::ConnectionError` (p. 1081), `activemq::commands::ConnectionInfo` (p. 1130), `activemq::commands::ConsumerControl` (p. 1167), `activemq::commands::ConsumerInfo` (p. 1217), `activemq::commands::ControlCommand` (p. 1244), `activemq::commands::DataArrayResponse` (p. 1269), `activemq::commands::DataResponse` (p. 1308), `activemq::commands::DestinationInfo` (p. 1392), `activemq::commands::DiscoveryEvent` (p. 1418), `activemq::commands::ExceptionResponse` (p. 1485), `activemq::commands::FlushCommand` (p. 1574), `activemq::commands::IntegerResponse` (p. 1668), `activemq::commands::JournalQueueAck` (p. 1719), `activemq::commands::JournalTopicAck` (p. 1743), `activemq::commands::JournalTrace` (p. 1767), `activemq::commands::JournalTransaction` (p. 1791), `activemq::commands::KeepAliveInfo` (p. 1813), `activemq::commands::LastPartialCommand` (p. 1841), `activemq::commands::Message` (p. 2023), `activemq::commands::MessageAck` (p. 2056), `activemq::commands::MessageDispatch` (p. 2085), `activemq::commands::MessageDispatchNotification` (p. 2117), `activemq::commands::MessagePull` (p. 2204), `activemq::commands::NetworkBridgeFilter` (p. 2247), `activemq::commands::PartialCommand` (p. 2320), `activemq::commands::ProducerAck` (p. 2421), `activemq::commands::ProducerInfo` (p. 2471), `activemq::commands::RemoveInfo` (p. 2538), `activemq::commands::RemoveSubscriptionInfo` (p. 2561), `activemq::commands::ReplayCommand` (p. 2585), `activemq::commands::Response` (p. 2612), `activemq::commands::SessionInfo` (p. 2703), `activemq::commands::ShutdownInfo` (p. 2758), `activemq::commands::SubscriptionInfo` (p. 2908), `activemq::commands::TransactionInfo` (p. 3038), and `activemq::commands::WireFormatInfo` (p. 3156).

6.264.2.2 virtual void activemq::commands::DataStructure::copyDataStructure (const DataStructure * *src*) [pure virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Implemented in **activemq::commands::ActiveMQBlobMessage** (p. 143), **activemq::commands::ActiveMQBytesMessage** (p. 170), **activemq::commands::ActiveMQDestination** (p. 243), **activemq::commands::ActiveMQMapMessage** (p. 276), **activemq::commands::ActiveMQMessage** (p. 306), **activemq::commands::ActiveMQObjectMessage** (p. 346), **activemq::commands::ActiveMQStreamMessage** (p. 425), **activemq::commands::ActiveMQTextMessage** (p. 527), **activemq::commands::BaseCommand** (p. 598), **activemq::commands::BrokerError** (p. 687), **activemq::commands::BrokerInfo** (p. 716), **activemq::commands::ConnectionControl** (p. 1057), **activemq::commands::ConnectionError** (p. 1081), **activemq::commands::ConnectionInfo** (p. 1130), **activemq::commands::ConsumerControl** (p. 1167), **activemq::commands::ConsumerInfo** (p. 1217), **activemq::commands::ControlCommand** (p. 1244), **activemq::commands::DataArrayResponse** (p. 1269), **activemq::commands::DataResponse** (p. 1308), **activemq::commands::DestinationInfo** (p. 1392), **activemq::commands::DiscoveryEvent** (p. 1418), **activemq::commands::ExceptionResponse** (p. 1485), **activemq::commands::FlushCommand** (p. 1574), **activemq::commands::IntegerResponse** (p. 1668), **activemq::commands::JournalQueueAck** (p. 1720), **activemq::commands::JournalTopicAck** (p. 1743), **activemq::commands::JournalTrace** (p. 1768), **activemq::commands::JournalTransaction** (p. 1791), **activemq::commands::KeepAliveInfo** (p. 1814), **activemq::commands::LastPartialCommand** (p. 1841), **activemq::commands::Message** (p. 2023), **activemq::commands::MessageAck** (p. 2057), **activemq::commands::MessageDispatch** (p. 2085), **activemq::commands::MessageDispatchNotification** (p. 2117), **activemq::commands::MessagePull** (p. 2204), **activemq::commands::NetworkBridgeFilter** (p. 2248), **activemq::commands::PartialCommand** (p. 2320), **activemq::commands::ProducerAck** (p. 2422), **activemq::commands::ProducerInfo** (p. 2471), **activemq::commands::RemoveInfo** (p. 2538), **activemq::commands::RemoveSubscriptionInfo** (p. 2562), **activemq::commands::ReplayCommand** (p. 2585), **activemq::commands::Response** (p. 2612), **activemq::commands::SessionInfo** (p. 2703), **activemq::commands::ShutdownInfo** (p. 2758), **activemq::commands::SubscriptionInfo** (p. 2909), **activemq::commands::TransactionInfo** (p. 3038), and **activemq::commands::WireFormatInfo** (p. 3156).

6.264.2.3 virtual bool activemq::commands::DataStructure::equals (const DataStructure * *value*) const [pure virtual]

Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Implemented in **activemq::commands::ActiveMQBlobMessage** (p. 144), **activemq::commands::ActiveMQBytesMessage** (p. 171), **activemq::commands::ActiveMQDestination** (p. 243), **activemq::commands::ActiveMQMapMessage** (p. 276), **activemq::commands::ActiveMQMessage** (p. 306), **activemq::commands::ActiveMQMessageTemplate<T>** (p. 331), **activemq::commands::ActiveMQObjectMessage** (p. 346), **activemq::commands::ActiveMQStreamMessage** (p. 426), **activemq::commands::ActiveMQTextMessage** (p. 527), **activemq::commands::BaseCommand** (p. 598), **activemq::commands::BooleanExpression** (p. 680), **activemq::commands::BrokerInfo** (p. 716), **activemq::commands::ConnectionControl** (p. 1057), **activemq::commands::ConnectionError** (p. 1081), **activemq::commands::ConnectionInfo** (p. 1131), **activemq::commands::ConsumerControl** (p. 1168), **activemq::commands::ConsumerInfo** (p. 1217), **activemq::commands::ControlCommand** (p. 1245), **activemq::commands::DataArrayResponse** (p. 1269), **activemq::commands::DataResponse** (p. 1309), **activemq::commands::DestinationInfo** (p. 1392), **activemq::commands::DiscoveryEvent** (p. 1418), **activemq::commands::ExceptionResponse** (p. 1485), **activemq::commands::FlushCommand** (p. 1574), **activemq::commands::IntegerResponse** (p. 1668), **activemq::commands::JournalQueueAck** (p. 1720), **activemq::commands::JournalTopicAck** (p. 1743), **activemq::commands::JournalTrace** (p. 1768), **activemq::commands::JournalTransaction** (p. 1791), **activemq::commands::KeepAliveInfo** (p. 1814), **activemq::commands::LastPartialCommand** (p. 1841), **activemq::commands::Message** (p. 2023), **activemq::commands::MessageAck** (p. 2057), **activemq::commands::MessageDispatch** (p. 2086), **activemq::commands::MessageDispatchNotification** (p. 2117), **activemq::commands::MessagePull** (p. 2204), **activemq::commands::NetworkBridgeFilter** (p. 2248), **activemq::commands::PartialCommand** (p. 2320), **activemq::commands::ProducerAck** (p. 2422), **activemq::commands::ProducerInfo** (p. 2471), **activemq::commands::RemoveInfo** (p. 2538), **activemq::commands::RemoveSubscriptionInfo** (p. 2562), **activemq::commands::ReplayCommand** (p. 2586), **activemq::commands::Response** (p. 2613), **activemq::commands::SessionInfo** (p. 2703), **activemq::commands::ShutdownInfo** (p. 2758), **activemq::commands::SubscriptionInfo** (p. 2909), **activemq::commands::TransactionInfo** (p. 3039), **activemq::commands::WireFormatInfo** (p. 3156), **activemq::commands::ActiveMQMessageTemplate<cms::BytesMessage>** (p. 331), **activemq::commands::ActiveMQMessageTemplate<cms::MapMessage>** (p. 331), **activemq::commands::ActiveMQMessageTemplate<cms::Message>** (p. 331), **activemq::commands::ActiveMQMessageTemplate<cms::StreamMessage>** (p. 331), **activemq::commands::ActiveMQMessageTemplate<cms::TextMessage>** (p. 331), and **activemq::commands::ActiveMQMessageTemplate<cms::ObjectMessage>** (p. 331).

6.264.2.4 virtual unsigned char activemq::commands::DataStructure::getDataStructureType () const [pure virtual]

Get the **DataStructure** (p. 1372) Type as defined in **CommandTypes.h**.

Returns

The type of the data structure

Implemented in `activemq::commands::ActiveMQBlobMessage` (p. 144), `activemq::commands::ActiveMQBytesMessage` (p. 172), `activemq::commands::ActiveMQDestination` (p. 244), `activemq::commands::ActiveMQMapMessage` (p. 278), `activemq::commands::ActiveMQMessage` (p. 306), `activemq::commands::ActiveMQObjectMessage` (p. 346), `activemq::commands::ActiveMQStreamMessage` (p. 426), `activemq::commands::ActiveMQTextMessage` (p. 528), `activemq::commands::BrokerError` (p. 688), `activemq::commands::BrokerInfo` (p. 717), `activemq::commands::ConnectionControl` (p. 1057), `activemq::commands::ConnectionError` (p. 1082), `activemq::commands::ConnectionInfo` (p. 1131), `activemq::commands::ConsumerControl` (p. 1168), `activemq::commands::ConsumerInfo` (p. 1218), `activemq::commands::ControlCommand` (p. 1245), `activemq::commands::DataArrayResponse` (p. 1270), `activemq::commands::DataResponse` (p. 1309), `activemq::commands::DestinationInfo` (p. 1393), `activemq::commands::DiscoveryEvent` (p. 1419), `activemq::commands::ExceptionResponse` (p. 1486), `activemq::commands::FlushCommand` (p. 1575), `activemq::commands::IntegerResponse` (p. 1669), `activemq::commands::JournalQueueAck` (p. 1720), `activemq::commands::JournalTopicAck` (p. 1744), `activemq::commands::JournalTrace` (p. 1768), `activemq::commands::JournalTransaction` (p. 1792), `activemq::commands::KeepAliveInfo` (p. 1814), `activemq::commands::LastPartialCommand` (p. 1842), `activemq::commands::Message` (p. 2025), `activemq::commands::MessageAck` (p. 2057), `activemq::commands::MessageDispatch` (p. 2086), `activemq::commands::MessageDispatchNotification` (p. 2118), `activemq::commands::MessagePull` (p. 2205), `activemq::commands::NetworkBridgeFilter` (p. 2248), `activemq::commands::PartialCommand` (p. 2321), `activemq::commands::ProducerAck` (p. 2422), `activemq::commands::ProducerInfo` (p. 2472), `activemq::commands::RemoveInfo` (p. 2538), `activemq::commands::RemoveSubscriptionInfo` (p. 2562), `activemq::commands::ReplayCommand` (p. 2586), `activemq::commands::Response` (p. 2613), `activemq::commands::SessionInfo` (p. 2704), `activemq::commands::ShutdownInfo` (p. 2758), `activemq::commands::SubscriptionInfo` (p. 2909), `activemq::commands::TransactionInfo` (p. 3039), and `activemq::commands::WireFormatInfo` (p. 3156).

6.264.2.5 `virtual std::string activemq::commands::DataStructure::toString ()`
`const [pure virtual]`

Returns a string containing the information for this **DataStructure** (p.1372) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Implemented in `activemq::commands::ActiveMQBlobMessage` (p. 146), `activemq::commands::ActiveMQBytesMessage` (p. 177), `activemq::commands::ActiveMQDestination` (p. 248), `activemq::commands::ActiveMQMapMessage` (p. 284), `activemq::commands::ActiveMQMessage` (p. 307), `activemq::commands::ActiveMQObjectMessage`

(p. 346), `activemq::commands::ActiveMQStreamMessage` (p. 432),
`activemq::commands::ActiveMQTextMessage` (p. 529), `ac-`
`tivemq::commands::BaseCommand` (p. 602), `activemq::commands::BaseDataStructure`
(p. 662), `activemq::commands::BooleanExpression` (p. 680), `ac-`
`tivemq::commands::BrokerInfo` (p. 719), `activemq::commands::Command`
(p. 995), `activemq::commands::ConnectionControl` (p. 1058), `ac-`
`tivemq::commands::ConnectionError` (p. 1082), `activemq::commands::ConnectionInfo`
(p. 1133), `activemq::commands::ConsumerControl` (p. 1169), `ac-`
`tivemq::commands::ConsumerInfo` (p. 1220), `activemq::commands::ControlCommand`
(p. 1245), `activemq::commands::DataArrayResponse` (p. 1270), `ac-`
`tivemq::commands::DataResponse` (p. 1309), `activemq::commands::DestinationInfo`
(p. 1394), `activemq::commands::DiscoveryEvent` (p. 1419),
`activemq::commands::ExceptionResponse` (p. 1486), `ac-`
`tivemq::commands::FlushCommand` (p. 1575), `activemq::commands::IntegerResponse`
(p. 1669), `activemq::commands::JournalQueueAck` (p. 1721), `ac-`
`tivemq::commands::JournalTopicAck` (p. 1745), `activemq::commands::JournalTrace`
(p. 1769), `activemq::commands::JournalTransaction` (p. 1792), `ac-`
`tivemq::commands::KeepAliveInfo` (p. 1815), `activemq::commands::LastPartialCommand`
(p. 1842), `activemq::commands::Message` (p. 2033), `activemq::commands::MessageAck`
(p. 2059), `activemq::commands::MessageDispatch` (p. 2087), `ac-`
`tivemq::commands::MessageDispatchNotification` (p. 2119), `ac-`
`tivemq::commands::MessagePull` (p. 2206), `activemq::commands::NetworkBridgeFilter`
(p. 2249), `activemq::commands::PartialCommand` (p. 2321), `ac-`
`tivemq::commands::ProducerAck` (p. 2423), `activemq::commands::ProducerInfo`
(p. 2473), `activemq::commands::RemoveInfo` (p. 2539), `ac-`
`tivemq::commands::RemoveSubscriptionInfo` (p. 2563), `ac-`
`tivemq::commands::ReplayCommand` (p. 2586), `activemq::commands::Response`
(p. 2614), `activemq::commands::SessionInfo` (p. 2704), `ac-`
`tivemq::commands::ShutdownInfo` (p. 2759), `activemq::commands::SubscriptionInfo`
(p. 2910), `activemq::commands::TransactionInfo` (p. 3040), and `ac-`
`tivemq::commands::WireFormatInfo` (p. 3162).

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/DataStructure.h`

6.265 decaf::util::Date Class Reference

Wrapper class around a time value in milliseconds.

```
#include <src/main/decaf/util/Date.h>
```

Inheritance diagram for `decaf::util::Date`:

Public Member Functions

- **Date** ()
Default constructor - sets time to the current System time, rounded to the nearest millisecond.
- **Date** (long long milliseconds)
Constructs the date with a given time value.

- **Date** (const **Date** &source)
Copy constructor.
- **Date** & **operator=** (const **Date** &value)
*Assigns the value of one **Date** (p. 1377) object to another.*
- virtual ~**Date** ()
- long long **getTime** () const
Gets the underlying time.
- void **setTime** (long long milliseconds)
Sets the underlying time.
- bool **after** (const **Date** &when) const
Determines whether or not this date falls after the specified time.
- bool **before** (const **Date** &when) const
Determines whether or not this date falls before the specified time.
- std::string **toString** () const
*Converts this **Date** (p. 1377) object to a String of the form:*
- virtual int **compareTo** (const **Date** &value) const
Compares this Data object to the one given.
- virtual bool **equals** (const **Date** &value) const
- virtual bool **operator==** (const **Date** &value) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const **Date** &value) const
Compares this object to another and returns true if this object is considered to be less than the one passed.

6.265.1 Detailed Description

Wrapper class around a time value in milliseconds. This class is comparable to Java's java.util.Date class.

Since

1.0

6.265.2 Constructor & Destructor Documentation

6.265.2.1 decaf::util::Date::Date ()

Default constructor - sets time to the current System time, rounded to the nearest millisecond.

6.265.2.2 decaf::util::Date::Date (long long *milliseconds*)

Constructs the date with a given time value.

Parameters

milliseconds The time in milliseconds;

6.265.2.3 decaf::util::Date::Date (const Date & *source*)

Copy constructor.

Parameters

source The **Date** (p. 1377) instance to copy into this one.

6.265.2.4 virtual decaf::util::Date::~~Date () [virtual]**6.265.3 Member Function Documentation****6.265.3.1 bool decaf::util::Date::after (const Date & *when*) const**

Determines whether or not this date falls after the specified time.

Parameters

when The date to compare

Returns

true if this date falls after when.

6.265.3.2 bool decaf::util::Date::before (const Date & *when*) const

Determines whether or not this date falls before the specified time.

Parameters

when The date to compare

Returns

true if this date falls before when.

6.265.3.3 virtual int decaf::util::Date::compareTo (const Date & *value*) const [virtual]

Compares this Data object to the one given.

Parameters

value The **Date** (p. 1377) value to compare to this one.

Returns

zero if the **Date** (p. 1377) values are equal, a value less than zero if this **Date** value is earlier than argument value, and a value greater than zero if this **Date** (p. 1377) object is later than the argument **Date** (p. 1377) value.

6.265.3.4 `virtual bool decaf::util::Date::equals (const Date & value) const`
[virtual]

Returns

true if this value is considered equal to the passed value.

6.265.3.5 `long long decaf::util::Date::getTime () const`

Gets the underlying time.

Returns

The underlying time value in milliseconds.

6.265.3.6 `virtual bool decaf::util::Date::operator< (const Date & value) const`
[virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

Parameters

value - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

6.265.3.7 `Date& decaf::util::Date::operator= (const Date & value)`

Assigns the value of one **Date** (p. 1377) object to another.

Parameters

value The value to be copied into this **Date** (p. 1377) object.

Returns

reference to this object with the newly assigned value.

6.265.3.8 virtual bool decaf::util::Date::operator==(const Date & *value*) const [virtual]

Compares equality between this object and the one passed.

Parameters

value - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

6.265.3.9 void decaf::util::Date::setTime (long long *milliseconds*)

Sets the underlying time.

Parameters

milliseconds The underlying time value in milliseconds.

6.265.3.10 std::string decaf::util::Date::toString () const

Converts this **Date** (p. 1377) object to a String of the form:

dow mon dd hh:mm:ss zzz yyyy

where:

- dow is the day of the week (Sun, Mon, Tue, Wed, Thu, Fri, Sat).
- mon is the month (Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec).
- dd is the day of the month (01 through 31), as two decimal digits.
- hh is the hour of the day (00 through 23), as two decimal digits.
- mm is the minute within the hour (00 through 59), as two decimal digits.
- ss is the second within the minute (00 through 61, as two decimal digits.
- zzz is the time zone (and may reflect daylight saving time). Standard time zone abbreviations include those recognized by the method parse. If time zone information is not available, then zzz is empty - that is, it consists of no characters at all.
- yyyy is the year, as four decimal digits.

Returns

the String representation of the **Date** (p. 1377) object.

The documentation for this class was generated from the following file:

- src/main/decaf/util/**Date.h**

6.266 decaf::internal::DecafRuntime Class Reference

Handles APR initialization and termination.

```
#include <src/main/decaf/internal/DecafRuntime.h>
```

Inheritance diagram for decaf::internal::DecafRuntime:

Public Member Functions

- **DecafRuntime** ()
Initializes the APR Runtime for a library.
- virtual **~DecafRuntime** ()
Terminates the APR Runtime for a library.
- apr_pool_t * **getGlobalPool** () const
Grants access to the Global APR Pool instance that should be used when creating new threads.

6.266.1 Detailed Description

Handles APR initialization and termination.

6.266.2 Constructor & Destructor Documentation

6.266.2.1 decaf::internal::DecafRuntime::DecafRuntime ()

Initializes the APR Runtime for a library.

6.266.2.2 virtual decaf::internal::DecafRuntime::~~DecafRuntime () [virtual]

Terminates the APR Runtime for a library.

6.266.3 Member Function Documentation

6.266.3.1 apr_pool_t* decaf::internal::DecafRuntime::getGlobalPool () const

Grants access to the Global APR Pool instance that should be used when creating new threads.

The documentation for this class was generated from the following file:

- src/main/decaf/internal/**DecafRuntime.h**

6.267 activemq::threads::DedicatedTaskRunner Class Reference

```
#include <src/main/activemq/threads/DedicatedTaskRunner.h>
```

Inheritance diagram for activemq::threads::DedicatedTaskRunner:

Public Member Functions

- **DedicatedTaskRunner** (**Task** *task)
- virtual **~DedicatedTaskRunner** ()
- virtual void **shutdown** (unsigned int timeout)

Shutdown after a timeout, does not guarantee that the task's iterate method has completed and the thread halted.
- virtual void **shutdown** ()

Shutdown once the task has finished and the TaskRunner's thread has exited.
- virtual void **wakeup** ()

*Signal the **TaskRunner** (p. 2958) to wakeup and execute another iteration cycle on the task, the **Task** (p. 2956) instance will be run until its iterate method has returned false indicating it is done.*

Protected Member Functions

- virtual void **run** ()

Run method - called by the Thread class in the context of the thread.

6.267.1 Constructor & Destructor Documentation

6.267.1.1 **activemq::threads::DedicatedTaskRunner::DedicatedTaskRunner** (**Task** * *task*)

6.267.1.2 **virtual activemq::threads::DedicatedTaskRunner::~~DedicatedTaskRunner** () [virtual]

6.267.2 Member Function Documentation

6.267.2.1 **virtual void activemq::threads::DedicatedTaskRunner::run** ()
[protected, virtual]

Run method - called by the Thread class in the context of the thread.

Implements **decaf::lang::Runnable** (p. 2642).

6.267.2.2 virtual void activemq::threads::DedicatedTaskRunner::shutdown () [virtual]

Shutdown once the task has finished and the TaskRunner's thread has exited.

6.267.2.3 virtual void activemq::threads::DedicatedTaskRunner::shutdown (unsigned int *timeout*) [virtual]

Shutdown after a timeout, does not guarantee that the task's iterate method has completed and the thread halted.

Parameters

timeout - Time in Milliseconds to wait for the task to stop.

6.267.2.4 virtual void activemq::threads::DedicatedTaskRunner::wakeup () [virtual]

Signal the **TaskRunner** (p. 2958) to wakeup and execute another iteration cycle on the task, the **Task** (p. 2956) instance will be run until its iterate method has returned false indicating it is done.

The documentation for this class was generated from the following file:

- src/main/activemq/threads/**DedicatedTaskRunner.h**

6.268 activemq::transport::DefaultTransportListener Class Reference

```
#include <src/main/activemq/transport/DefaultTransportListener.h>
```

Inheritance diagram for activemq::transport::DefaultTransportListener:

Public Member Functions

- virtual **~DefaultTransportListener** ()
- virtual void **onCommand** (const **Pointer**< **Command** > &command AMQCPP_UNUSED)
Event handler for the receipt of a command.
- virtual void **onException** (const **decaf::lang::Exception** &ex AMQCPP_UNUSED)
Event handler for an exception from a command transport.
- virtual void **transportInterrupted** ()
The transport has suffered an interruption from which it hopes to recover.
- virtual void **transportResumed** ()
The transport has resumed after an interruption.

6.268.1 Constructor & Destructor Documentation

6.268.1.1 virtual
activemq::transport::DefaultTransportListener::~~DefaultTransportListener
() [inline, virtual]

6.268.2 Member Function Documentation

6.268.2.1 virtual void activemq::transport::DefaultTransportListener::onCommand
(const Pointer< Command > &command *AMQCPP_UNUSED*)
[inline, virtual]

Event handler for the receipt of a command.

The transport passes off all received commands to its listeners, the listener then owns the Object. If there is no registered listener the **Transport** (p. 3066) deletes the command upon receipt.

Parameters

command the received command object.

6.268.2.2 virtual void activemq::transport::DefaultTransportListener::onException
(const decaf::lang::Exception &ex *AMQCPP_UNUSED*) [inline, virtual]

Event handler for an exception from a command transport.

Parameters

ex The exception.

6.268.2.3 virtual void activemq::transport::DefaultTransportListener::transportInterrupted ()
[inline, virtual]

The transport has suffered an interruption from which it hopes to recover.

Implements **activemq::transport::TransportListener** (p. 3082).

6.268.2.4 virtual void activemq::transport::DefaultTransportListener::transportResumed ()
[inline, virtual]

The transport has resumed after an interruption.

Implements **activemq::transport::TransportListener** (p. 3082).

The documentation for this class was generated from the following file:

- src/main/activemq/transport/**DefaultTransportListener.h**

6.269 decaf::util::concurrent::Delayed Class Reference

A mix-in style interface for marking objects that should be acted upon after a given delay.

```
#include <src/main/decaf/util/concurrent/Delayed.h>
```

Inheritance diagram for decaf::util::concurrent::Delayed:

Public Member Functions

- virtual `~Delayed()`
- virtual long long `getDelay(const TimeUnit &unit)=0`
Returns the remaining delay associated with this object, in the given time unit.

6.269.1 Detailed Description

A mix-in style interface for marking objects that should be acted upon after a given delay. An implementation of this interface must define a Comparable methods that provides an ordering consistent with its getDelay method.

6.269.2 Constructor & Destructor Documentation

6.269.2.1 virtual decaf::util::concurrent::Delayed::~~Delayed () [inline, virtual]

6.269.3 Member Function Documentation

6.269.3.1 virtual long long decaf::util::concurrent::Delayed::getDelay (const TimeUnit & *unit*) [pure virtual]

Returns the remaining delay associated with this object, in the given time unit.

Parameters

unit The time unit

Returns

the remaining delay; zero or negative values indicate that the delay has already elapsed

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/Delayed.h

6.270 cms::DeliveryMode Class Reference

This is an Abstract class whose purpose is to provide a container for the delivery mode enumeration for CMS messages.

```
#include <src/main/cms/DeliveryMode.h>
```

Public Types

- enum **DELIVERY_MODE** { **PERSISTENT** = 0, **NON_PERSISTENT** = 1 }
*Enumeration values for **Message** (p. 2036) Delivery Mode.*

Public Member Functions

- virtual **~DeliveryMode** ()

6.270.1 Detailed Description

This is an Abstract class whose purpose is to provide a container for the delivery mode enumeration for CMS messages. When a client sends a **cms::Message** (p. 2036) it can mark the **Message** (p. 2036) as either Persistent or Non-Persistent. If the client feels that the **Message** (p. 2036) cannot be lost in transit it should mark it as Persistent, otherwise if it is allowable for a **Message** (p. 2036) to occasionally be lost it can mark it as Non-Persistent. This allows the Provider to balance make tradeoffs between balance and **Message** (p. 2036) throughput.

The **DeliveryMode** (p. 1386) covers only the transport of the **Message** (p. 2036) for sending client to its destination and doesn't apply to the receiving **Message** (p. 2036) consumer. The receiving Consumer can drop Message's based on configuration such as memory limits or **Message** (p. 2036) filtering.

A message is guaranteed to be delivered once and only once by a CMS provider if the delivery mode of the message is **PERSISTENT** and the configuration of the **Message** (p. 2036) consumer allows for it.

Since

1.0

6.270.2 Member Enumeration Documentation

6.270.2.1 enum cms::DeliveryMode::DELIVERY_MODE

Enumeration values for **Message** (p. 2036) Delivery Mode.

Enumerator:

PERSISTENT

NON_PERSISTENT

6.270.3 Constructor & Destructor Documentation

6.270.3.1 virtual cms::DeliveryMode::~~DeliveryMode () [inline, virtual]

The documentation for this class was generated from the following file:

- src/main/cms/**DeliveryMode.h**

6.271 cms::Destination Class Reference

A **Destination** (p. 1387) object encapsulates a provider-specific address.

```
#include <src/main/cms/Destination.h>
```

Inheritance diagram for cms::Destination:

Public Types

- enum **DestinationType** { **TOPIC**, **QUEUE**, **TEMPORARY_TOPIC**, **TEMPORARY_QUEUE** }

Public Member Functions

- virtual **~Destination** ()
- virtual **DestinationType** **getDestinationType** () const =0
*Retrieve the **Destination** (p. 1387) Type for this **Destination** (p. 1387).*
- virtual **cms::Destination * clone** () const =0
Creates a new instance of this destination type that is a copy of this one, and returns it.
- virtual void **copy** (const **cms::Destination** &source)=0
*Copies the contents of the given **Destination** (p. 1387) object to this one.*
- virtual const **CMSProperties** & **getCMSProperties** () const =0
Retrieve any properties that might be part of the destination that was specified.

6.271.1 Detailed Description

A **Destination** (p. 1387) object encapsulates a provider-specific address. There is no standard definition of a **Destination** (p. 1387) address, each provider can provide its own definition and there can be configuration data attached to the **Destination** (p. 1387) address.

All CMS **Destination** (p. 1387) objects support concurrent use.

Since

1.0

6.271.2 Member Enumeration Documentation

6.271.2.1 enum cms::Destination::DestinationType

Enumerator:

TOPIC
QUEUE
TEMPORARY_TOPIC
TEMPORARY_QUEUE

6.271.3 Constructor & Destructor Documentation

6.271.3.1 virtual cms::Destination::~~Destination () [inline, virtual]

6.271.4 Member Function Documentation

6.271.4.1 virtual cms::Destination* cms::Destination::clone () const [pure virtual]

Creates a new instance of this destination type that is a copy of this one, and returns it.

Returns

cloned copy of this object

Implemented in **activemq::commands::ActiveMQQueue** (p. 380),
activemq::commands::ActiveMQTempQueue (p. 478), **ac-**
tivemq::commands::ActiveMQTempTopic (p. 502), and **ac-**
tivemq::commands::ActiveMQTopic (p. 551).

6.271.4.2 virtual void cms::Destination::copy (const cms::Destination & *source*)
[pure virtual]

Copies the contents of the given **Destination** (p. 1387) object to this one.

Parameters

source The source **Destination** (p. 1387) object.

Implemented in **activemq::commands::ActiveMQQueue** (p. 380),
activemq::commands::ActiveMQTempQueue (p. 479), **ac-**
tivemq::commands::ActiveMQTempTopic (p. 503), and **ac-**
tivemq::commands::ActiveMQTopic (p. 551).

6.271.4.3 virtual const CMSProperties& cms::Destination::getCMSProperties ()
const [pure virtual]

Retrieve any properties that might be part of the destination that was specified.

This is a deviation from the JMS spec but necessary due to C++ restrictions.

Returns

A {const} reference to a **CMSProperties** (p. 965) object.

Implemented in **activemq::commands::ActiveMQQueue** (p. 381),
activemq::commands::ActiveMQTempQueue (p. 480), **ac-**
tivemq::commands::ActiveMQTempTopic (p. 504), and **ac-**
tivemq::commands::ActiveMQTopic (p. 552).

6.271.4.4 virtual DestinationType cms::Destination::getDestinationType ()
const [pure virtual]

Retrieve the **Destination** (p. 1387) Type for this **Destination** (p. 1387).

Returns

The **Destination** (p.1387) Type

Implemented in **activemq::commands::ActiveMQQueue** (p. 382),
activemq::commands::ActiveMQTempQueue (p. 480), **ac-**
tivemq::commands::ActiveMQTempTopic (p. 504), and **ac-**
tivemq::commands::ActiveMQTopic (p. 552).

The documentation for this class was generated from the following file:

- `src/main/cms/Destination.h`

6.272 **activemq::commands::ActiveMQDestination::DestinationFilter** Struct Reference

```
#include <src/main/activemq/commands/ActiveMQDestination.h>
```

Static Public Attributes

- static const std::string **ANY_CHILD**
- static const std::string **ANY_DESCENDENT**

6.272.1 Field Documentation

6.272.1.1 const std::string
activemq::commands::ActiveMQDestination::DestinationFilter::ANY_
CHILD [static]

6.272.1.2 const std::string
activemq::commands::ActiveMQDestination::DestinationFilter::ANY_
DESCENDENT [static]

The documentation for this struct was generated from the following file:

- `src/main/activemq/commands/ActiveMQDestination.h`

6.273 **activemq::commands::DestinationInfo** Class Reference

```
#include <src/main/activemq/commands/DestinationInfo.h>
```

Inheritance diagram for **activemq::commands::DestinationInfo**:

Public Member Functions

- **DestinationInfo** ()
- virtual **~DestinationInfo** ()

- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **DestinationInfo** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1372) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **ConnectionId** > & **getConnectionId** () const
- virtual **Pointer**< **ConnectionId** > & **getConnectionId** ()
- virtual void **setConnectionId** (const **Pointer**< **ConnectionId** > &connectionId)
- virtual const **Pointer**< **ActiveMQDestination** > & **getDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &destination)
- virtual unsigned char **getOperationType** () const
- virtual void **setOperationType** (unsigned char operationType)
- virtual long long **getTimeout** () const
- virtual void **setTimeout** (long long timeout)
- virtual const std::vector< **decaf::lang::Pointer**< **BrokerId** > > & **getBrokerPath** () const
- virtual std::vector< **decaf::lang::Pointer**< **BrokerId** > > & **getBrokerPath** ()
- virtual void **setBrokerPath** (const std::vector< **decaf::lang::Pointer**< **BrokerId** > > &brokerPath)
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor *visitor) throw (exceptions::ActiveMQException)
Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_DESTINATIONINFO** = 8

Protected Member Functions

- **DestinationInfo** (const **DestinationInfo** &)
- **DestinationInfo** & **operator=** (const **DestinationInfo** &)

Protected Attributes

- `Pointer< ConnectionId > connectionId`
- `Pointer< ActiveMQDestination > destination`
- unsigned char `operationType`
- long long `timeout`
- `std::vector< decaf::lang::Pointer< BrokerId > > brokerPath`

6.273.1 Constructor & Destructor Documentation

- 6.273.1.1** `activemq::commands::DestinationInfo::DestinationInfo (const DestinationInfo &)` [inline, protected]
- 6.273.1.2** `activemq::commands::DestinationInfo::DestinationInfo ()`
- 6.273.1.3** `virtual activemq::commands::DestinationInfo::~~DestinationInfo ()` [virtual]

6.273.2 Member Function Documentation

- 6.273.2.1** `virtual DestinationInfo* activemq::commands::DestinationInfo::cloneDataStructure () const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1373).

- 6.273.2.2** `virtual void activemq::commands::DestinationInfo::copyDataStructure (const DataStructure * src)` [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

Reimplemented from `activemq::commands::BaseCommand` (p. 598).

- 6.273.2.3** `virtual bool activemq::commands::DestinationInfo::equals (const DataStructure * value) const` [virtual]

Compares the `DataStructure` (p. 1372) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 598).

- 6.273.2.4 **virtual const std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::DestinationInfo::getBrokerPath () const**
[virtual]
- 6.273.2.5 **virtual std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::DestinationInfo::getBrokerPath ()** [virtual]
- 6.273.2.6 **virtual Pointer<ConnectionId>& activemq::commands::DestinationInfo::getConnectionId ()** [virtual]
- 6.273.2.7 **virtual const Pointer<ConnectionId>& activemq::commands::DestinationInfo::getConnectionId () const** [virtual]
- 6.273.2.8 **virtual unsigned char activemq::commands::DestinationInfo::getDataStructureType () const**
[virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataStructure** (p. 1372) type copy.

Implements **activemq::commands::DataStructure** (p. 1375).

- 6.273.2.9 `virtual const Pointer<ActiveMQDestination>& activemq::commands::DestinationInfo::getDestination () const` [virtual]
- 6.273.2.10 `virtual Pointer<ActiveMQDestination>& activemq::commands::DestinationInfo::getDestination ()` [virtual]
- 6.273.2.11 `virtual unsigned char activemq::commands::DestinationInfo::getOperationType () const` [virtual]
- 6.273.2.12 `virtual long long activemq::commands::DestinationInfo::getTimeout () const` [virtual]
- 6.273.2.13 `DestinationInfo& activemq::commands::DestinationInfo::operator= (const DestinationInfo &)` [inline, protected]
- 6.273.2.14 `virtual void activemq::commands::DestinationInfo::setBrokerPath (const std::vector< decaf::lang::Pointer< BrokerId > > & brokerPath)` [virtual]
- 6.273.2.15 `virtual void activemq::commands::DestinationInfo::setConnectionId (const Pointer< ConnectionId > & connectionId)` [virtual]
- 6.273.2.16 `virtual void activemq::commands::DestinationInfo::setDestination (const Pointer< ActiveMQDestination > & destination)` [virtual]
- 6.273.2.17 `virtual void activemq::commands::DestinationInfo::setOperationType (unsigned char operationType)` [virtual]
- 6.273.2.18 `virtual void activemq::commands::DestinationInfo::setTimeout (long long timeout)` [virtual]
- 6.273.2.19 `virtual std::string activemq::commands::DestinationInfo::toString () const` [virtual]

Returns a string containing the information for this **DataStructure** (p.1372) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from `activemq::commands::BaseCommand` (p. 602).

- 6.273.2.20 `virtual Pointer<Command> activemq::commands::DestinationInfo::visit (activemq::state::CommandVisitor * visitor) throw (exceptions::ActiveMQException)` [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

Returns

a **Response** (p. 2611) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 995).

6.273.3 Field Documentation

- 6.273.3.1** `std::vector< decaf::lang::Pointer<BrokerId> >`
`activemq::commands::DestinationInfo::brokerPath` [protected]
- 6.273.3.2** `Pointer<ConnectionId>` `activemq::commands::DestinationInfo::connectionId`
[protected]
- 6.273.3.3** `Pointer<ActiveMQDestination>` `activemq::commands::DestinationInfo::destination`
[protected]
- 6.273.3.4** `const unsigned char` `activemq::commands::DestinationInfo::ID_ - DESTINATIONINFO = 8` [static]
- 6.273.3.5** `unsigned char` `activemq::commands::DestinationInfo::operationType`
[protected]
- 6.273.3.6** `long long` `activemq::commands::DestinationInfo::timeout` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/DestinationInfo.h`

6.274 activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1395).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/DestinationInfoMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller`:

Public Member Functions

- **DestinationInfoMarshaller** ()
- virtual **~DestinationInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.274.1 Detailed Description

Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1395). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.274.2 Constructor & Destructor Documentation

6.274.2.1 activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller::DestinationInfoMarshaller () [inline]

6.274.2.2 virtual activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller::~~DestinationInfoMarshaller () [inline, virtual]

6.274.3 Member Function Documentation

6.274.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1331).

6.274.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller::getDataStructureType() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1336).

6.274.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 631).

6.274.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 632).

```
6.274.3.5 virtual int ac-
      tivemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller::tightMarshal1
      ( OpenWireFormat * wireFormat, commands::DataStructure
        * dataStructure, utils::BooleanStream * bs ) throw (
        decaf::io::IOException ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 633).

```
6.274.3.6 virtual void ac-
      tivemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller::tightMarshal2
      ( OpenWireFormat * wireFormat, commands::DataStructure
        * dataStructure, decaf::io::DataOutputStream * dataOut,
        utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 634).

6.274.3.7 virtual void activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 635).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**DestinationInfoMarshaller.h**

6.275 activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1399).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/DestinationInfoMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller:

Public Member Functions

- **DestinationInfoMarshaller** ()
- virtual ~**DestinationInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

6.275.1 Detailed Description

Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p.1399). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.275.2 Constructor & Destructor Documentation

6.275.2.1 **activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller::DestinationInfoMarshaller** () [inline]

6.275.2.2 **virtual** **activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller::~~DestinationInfoMarshaller** () [inline, virtual]

6.275.3 Member Function Documentation

6.275.3.1 **virtual** **commands::DataStructure*** **activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller::createObject** () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1331).

6.275.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1336).

6.275.3.3 virtual void activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 604).

6.275.3.4 virtual void activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 605).

6.275.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 606).

6.275.3.6 `virtual void activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 608).

6.275.3.7 `virtual void activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 609).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/DestinationInfoMarshaller.h`

6.276 activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1403).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/DestinationInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller**:

Public Member Functions

- **DestinationInfoMarshaller** ()
- virtual **~DestinationInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.276.1 Detailed Description

Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p.1403). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.276.2 Constructor & Destructor Documentation

6.276.2.1 **activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller::DestinationInfoMarshaller** () [inline]

6.276.2.2 **virtual**
activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller::~~DestinationInfoMarshaller () [inline, virtual]

6.276.3 Member Function Documentation

6.276.3.1 **virtual commands::DataStructure*** **activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller::createObject** () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1331).

6.276.3.2 **virtual unsigned char** **activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller::getDataStructureType** () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1336).

6.276.3.3 virtual void activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 618).

6.276.3.4 virtual void activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 619).

6.276.3.5 virtual int activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 620).

```
6.276.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 621).

```
6.276.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 622).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/DestinationInfoMarshaller.h`

6.277 activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1407).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/DestinationInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller**:

Public Member Functions

- **DestinationInfoMarshaller** ()
- virtual **~DestinationInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.277.1 Detailed Description

Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1407). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.277.2 Constructor & Destructor Documentation

6.277.2.1 activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller::DestinationInfoMarshaller () [inline]

6.277.2.2 virtual
activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller::~~DestinationInfoMarshaller () [inline, virtual]

6.277.3 Member Function Documentation

6.277.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1331).

6.277.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1336).

6.277.3.3 virtual void activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 611).

6.277.3.4 virtual void activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 612).

6.277.3.5 virtual int activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 613).

```
6.277.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 614).

```
6.277.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 615).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/DestinationInfoMarshaller.h`

6.278 `activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller` Class Reference

Marshaling code for Open Wire Format for `DestinationInfoMarshaller` (p. 1411).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/DestinationInfoMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller`:

Public Member Functions

- `DestinationInfoMarshaller ()`
- `virtual ~DestinationInfoMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaller.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write a object instance to data output stream.
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.278.1 Detailed Description

Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1411). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.278.2 Constructor & Destructor Documentation

6.278.2.1 activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller::DestinationInfoMarshaller () [inline]

6.278.2.2 virtual
activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller::~~DestinationInfoMarshaller () [inline, virtual]

6.278.3 Member Function Documentation

6.278.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1331).

6.278.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1336).

6.278.3.3 virtual void activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 624).

6.278.3.4 virtual void activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 625).

6.278.3.5 virtual int activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 626).

```
6.278.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 628).

```
6.278.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 629).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/DestinationInfoMarshaller.h`

6.279 activemq::cmsutil::DestinationResolver Class Reference

Resolves a CMS destination name to a `Destination`.

```
#include <src/main/activemq/cmsutil/DestinationResolver.h>
```

Inheritance diagram for `activemq::cmsutil::DestinationResolver`:

Public Member Functions

- virtual `~DestinationResolver()`
- virtual void **init** (`ResourceLifecycleManager *mgr`)=0
Initializes this destination resolver for use.
- virtual void **destroy** ()=0
Destroys any allocated resources.
- virtual `cms::Destination * resolveDestinationName (cms::Session *session, const std::string &destName, bool pubSubDomain)=0` throw (`cms::CMSException`)
Resolves the given name to a destination.

6.279.1 Detailed Description

Resolves a CMS destination name to a `Destination`.

6.279.2 Constructor & Destructor Documentation

- 6.279.2.1** virtual `activemq::cmsutil::DestinationResolver::~DestinationResolver()` [`inline`, `virtual`]

6.279.3 Member Function Documentation

- 6.279.3.1** virtual void `activemq::cmsutil::DestinationResolver::destroy()` [`pure virtual`]

Destroys any allocated resources.

Implemented in `activemq::cmsutil::DynamicDestinationResolver` (p. 1472).

6.279.3.2 `virtual void activemq::cmsutil::DestinationResolver::init (ResourceLifecycleManager * mgr) [pure virtual]`

Initializes this destination resolver for use.

Ensures that any previously allocated resources are first destroyed (e.g. calls `destroy()` (p. 1415)).

Parameters

mgr the resource lifecycle manager.

Implemented in `activemq::cmsutil::DynamicDestinationResolver` (p. 1472).

6.279.3.3 `virtual cms::Destination* activemq::cmsutil::DestinationResolver::resolveDestinationName (cms::Session * session, const std::string & destName, bool pubSubDomain) throw (cms::CMSException) [pure virtual]`

Resolves the given name to a destination.

If `pubSubDomain` is true, a topic will be returned, otherwise a queue will be returned.

Parameters

session the session for which to retrieve resolve the destination.

destName the name to be resolved.

pubSubDomain If true, the name will be resolved to a Topic, otherwise a Queue.

Returns

the resolved destination

Exceptions

cms::CMSException (p. 960) if resolution failed.

Implemented in `activemq::cmsutil::DynamicDestinationResolver` (p. 1472).

The documentation for this class was generated from the following file:

- `src/main/activemq/cmsutil/DestinationResolver.h`

6.280 activemq::commands::DiscoveryEvent Class Reference

```
#include <src/main/activemq/commands/DiscoveryEvent.h>
```

Inheritance diagram for `activemq::commands::DiscoveryEvent`:

Public Member Functions

- **DiscoveryEvent** ()
- virtual **~DiscoveryEvent** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **DiscoveryEvent * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1372) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent.*
- virtual const std::string & **getServiceName** () const
- virtual std::string & **getServiceName** ()
- virtual void **setServiceName** (const std::string &serviceName)
- virtual const std::string & **getBrokerName** () const
- virtual std::string & **getBrokerName** ()
- virtual void **setBrokerName** (const std::string &brokerName)

Static Public Attributes

- static const unsigned char **ID_DISCOVERYEVENT** = 40

Protected Member Functions

- **DiscoveryEvent** (const **DiscoveryEvent** &)
- **DiscoveryEvent** & **operator=** (const **DiscoveryEvent** &)

Protected Attributes

- std::string **serviceName**
- std::string **brokerName**

6.280.1 Constructor & Destructor Documentation

6.280.1.1 `activemq::commands::DiscoveryEvent::DiscoveryEvent (const
DiscoveryEvent &) [inline, protected]`

6.280.1.2 `activemq::commands::DiscoveryEvent::DiscoveryEvent ()`

6.280.1.3 `virtual activemq::commands::DiscoveryEvent::~~DiscoveryEvent ()
[virtual]`

6.280.2 Member Function Documentation

6.280.2.1 `virtual DiscoveryEvent* ac-
tivemq::commands::DiscoveryEvent::cloneDataStructure () const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1373).

6.280.2.2 `virtual void activemq::commands::DiscoveryEvent::copyDataStructure (const
DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

Implements `activemq::commands::DataStructure` (p. 1374).

6.280.2.3 `virtual bool activemq::commands::DiscoveryEvent::equals (const
DataStructure * value) const [virtual]`

Compares the `DataStructure` (p. 1372) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Implements `activemq::commands::DataStructure` (p. 1374).

- 6.280.2.4 `virtual const std::string& activemq::commands::DiscoveryEvent::getBrokerName () const [virtual]`
- 6.280.2.5 `virtual std::string& activemq::commands::DiscoveryEvent::getBrokerName () [virtual]`
- 6.280.2.6 `virtual unsigned char activemq::commands::DiscoveryEvent::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataSet** (p. 1372) type copy.

Implements **activemq::commands::DataSet** (p. 1375).

- 6.280.2.7 `virtual std::string& activemq::commands::DiscoveryEvent::getServiceName () [virtual]`
- 6.280.2.8 `virtual const std::string& activemq::commands::DiscoveryEvent::getServiceName () const [virtual]`
- 6.280.2.9 `DiscoveryEvent& activemq::commands::DiscoveryEvent::operator= (const DiscoveryEvent &) [inline, protected]`
- 6.280.2.10 `virtual void activemq::commands::DiscoveryEvent::setBrokerName (const std::string & brokerName) [virtual]`
- 6.280.2.11 `virtual void activemq::commands::DiscoveryEvent::setServiceName (const std::string & serviceName) [virtual]`
- 6.280.2.12 `virtual std::string activemq::commands::DiscoveryEvent::toString () const [virtual]`

Returns a string containing the information for this **DataSet** (p. 1372) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataSet** (p. 662).

6.280.3 Field Documentation

- 6.280.3.1 `std::string activemq::commands::DiscoveryEvent::brokerName`
[protected]
- 6.280.3.2 `const unsigned char activemq::commands::DiscoveryEvent::ID _ - DISCOVERYEVENT = 40` [static]
- 6.280.3.3 `std::string activemq::commands::DiscoveryEvent::serviceName`
[protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/DiscoveryEvent.h`

6.281 `activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller` Class Reference

Marshaling code for Open Wire Format for `DiscoveryEventMarshaller` (p. 1420).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/DiscoveryEventMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller`:

Public Member Functions

- `DiscoveryEventMarshaller ()`
- `virtual ~DiscoveryEventMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaller.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write a object instance to data output stream.

- virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.281.1 Detailed Description

Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1420). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.281.2 Constructor & Destructor Documentation

6.281.2.1 `activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller::DiscoveryEventMarshaller()` [inline]

6.281.2.2 `virtual activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller::~~DiscoveryEventMarshaller()` [inline, virtual]

6.281.3 Member Function Documentation

6.281.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.281.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.281.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1342).

6.281.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1348).

6.281.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1354).

6.281.3.6 virtual void **activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller::tightMarshal2** (**OpenWireFormat** * *wireFormat*, **commands::DataStructure** * *dataStructure*, **decaf::io::DataOutputStream** * *dataOut*, **utils::BooleanStream** * *bs*) throw (**decaf::io::IOException**) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1360).

6.281.3.7 virtual void **activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller::tightUnmarshal** (**OpenWireFormat** * *wireFormat*, **commands::DataStructure** * *dataStructure*, **decaf::io::DataInputStream** * *dataIn*, **utils::BooleanStream** * *bs*) throw (**decaf::io::IOException**) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1366).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/DiscoveryEventMarshaller.h`

6.282 **activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller** Class Reference

Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1424).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/DiscoveryEventMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller**:

Public Member Functions

- **DiscoveryEventMarshaller** ()
- virtual **~DiscoveryEventMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.282.1 Detailed Description

Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p.1424). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.282.2 Constructor & Destructor Documentation

6.282.2.1 `activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller::DiscoveryEventMarshaller ()` [inline]

6.282.2.2 `virtual activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller::~~DiscoveryEventMarshaller ()` [inline, virtual]

6.282.3 Member Function Documentation

6.282.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller::createObject () const` [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p.1331).

6.282.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller::getDataStructureType () const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p.1336).

6.282.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1342).

```
6.282.3.4 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1348).

```
6.282.3.5 virtual int ac-
tivemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, utils::BooleanStream * bs ) throw (
decaf::io::IOException ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1354).

6.282.3.6 virtual void activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1360).

6.282.3.7 virtual void activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1366).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/DiscoveryEventMarshaller.h

6.283 activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller Class Reference

Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1427).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/DiscoveryEventMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller`:

Public Member Functions

- **DiscoveryEventMarshaller** ()
- virtual **~DiscoveryEventMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.283.1 Detailed Description

Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1427). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module

6.283.2 Constructor & Destructor Documentation

6.283.2.1 `activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller::DiscoveryEventMarshaller () [inline]`

6.283.2.2 `virtual activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller::~~DiscoveryEventMarshaller () [inline, virtual]`

6.283.3 Member Function Documentation

6.283.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.283.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.283.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1342).

6.283.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1348).

6.283.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1354).

6.283.3.6 `virtual void activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1360).

6.283.3.7 `virtual void activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1366).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/DiscoveryEventMarshaller.h`

6.284 `activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller` Class Reference

Marshaling code for Open Wire Format for `DiscoveryEventMarshaller` (p. 1431).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/DiscoveryEventMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller`:

Public Member Functions

- **DiscoveryEventMarshaller** ()
- virtual **~DiscoveryEventMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.284.1 Detailed Description

Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p.1431). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.284.2 Constructor & Destructor Documentation

6.284.2.1 `activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller::DiscoveryEventMarshaller () [inline]`

6.284.2.2 `virtual activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller::~~DiscoveryEventMarshaller () [inline, virtual]`

6.284.3 Member Function Documentation

6.284.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.284.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.284.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1342).

6.284.3.4 virtual void activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1348).

6.284.3.5 virtual int activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1354).

6.284.3.6 virtual void activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1360).

6.284.3.7 virtual void activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1366).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/DiscoveryEventMarshaller.h

6.285 activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller Class Reference

Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1435).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/DiscoveryEventMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller:

Public Member Functions

- **DiscoveryEventMarshaller** ()
- virtual **~DiscoveryEventMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.285.1 Detailed Description

Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1435). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.285.2 Constructor & Destructor Documentation

6.285.2.1 `activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller::DiscoveryEventMarshaller()` [inline]

6.285.2.2 `virtual activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller::~~DiscoveryEventMarshaller()` [inline, virtual]

6.285.3 Member Function Documentation

6.285.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.285.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.285.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1342).

6.285.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1348).

6.285.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1354).

6.285.3.6 `virtual void activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions

- IOException* if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1360).

6.285.3.7 virtual void activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller::tightUnmarshal
 (**OpenWireFormat** * *wireFormat*, **commands::DataStructure**
 * *dataStructure*, **decaf::io::DataInputStream** * *dataIn*,
utils::BooleanStream * *bs*) throw (**decaf::io::IOException**) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1366).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/DiscoveryEventMarshaller.h`

6.286 activemq::core::DispatchData Class Reference

Simple POCO that contains the information necessary to route a message to a specified consumer.

```
#include <src/main/activemq/core/DispatchData.h>
```

Public Member Functions

- **DispatchData** ()
- **DispatchData** (const **decaf::lang::Pointer**< **commands::ConsumerId** > &consumer,
const **decaf::lang::Pointer**< **commands::Message** > &message)
- const **decaf::lang::Pointer**< **commands::ConsumerId** > & **getConsumerId** ()
- const **decaf::lang::Pointer**< **commands::Message** > & **getMessage** ()

6.286.1 Detailed Description

Simple POCO that contains the information necessary to route a message to a specified consumer.

6.286.2 Constructor & Destructor Documentation

6.286.2.1 `activemq::core::DispatchData::DispatchData ()` [inline]

6.286.2.2 `activemq::core::DispatchData::DispatchData (const decaf::lang::Pointer< commands::ConsumerId > & consumer, const decaf::lang::Pointer< commands::Message > & message)` [inline]

6.286.3 Member Function Documentation

6.286.3.1 `const decaf::lang::Pointer<commands::ConsumerId>& activemq::core::DispatchData::getConsumerId ()` [inline]

6.286.3.2 `const decaf::lang::Pointer<commands::Message>& activemq::core::DispatchData::getMessage ()` [inline]

The documentation for this class was generated from the following file:

- `src/main/activemq/core/DispatchData.h`

6.287 activemq::core::Dispatcher Class Reference

Interface for an object responsible for dispatching messages to consumers.

```
#include <src/main/activemq/core/Dispatcher.h>
```

Inheritance diagram for `activemq::core::Dispatcher`:

Public Member Functions

- virtual `~Dispatcher ()`
- virtual void `dispatch (const Pointer< MessageDispatch > &message)=0`

Dispatches a message to a particular consumer.

6.287.1 Detailed Description

Interface for an object responsible for dispatching messages to consumers.

6.287.2 Constructor & Destructor Documentation

6.287.2.1 virtual `activemq::core::Dispatcher::~~Dispatcher ()` [inline, virtual]

6.287.3 Member Function Documentation

6.287.3.1 virtual void `activemq::core::Dispatcher::dispatch (const Pointer< MessageDispatch > & message)` [pure virtual]

Dispatches a message to a particular consumer.

Parameters

message - the message to be dispatched.

Implemented in `activemq::core::ActiveMQConsumer` (p. 234), and `activemq::core::ActiveMQSession` (p. 413).

The documentation for this class was generated from the following file:

- `src/main/activemq/core/Dispatcher.h`

6.288 decaf::lang::Double Class Reference

```
#include <src/main/decaf/lang/Double.h>
```

Inheritance diagram for `decaf::lang::Double`:

Public Member Functions

- **Double** (double value)
- **Double** (const std::string &value) throw (exceptions::NumberFormatException)
- virtual **~Double** ()
- virtual int **compareTo** (const **Double** &d) const
*Compares this **Double** (p. 1441) instance with another.*
- bool **equals** (const **Double** &d) const
- virtual bool **operator==** (const **Double** &d) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const **Double** &d) const
Compares this object to another and returns true if this object is considered to be less than the one passed.
- virtual int **compareTo** (const double &d) const
*Compares this **Double** (p. 1441) instance with another.*
- bool **equals** (const double &d) const
- virtual bool **operator==** (const double &d) const

Compares equality between this object and the one passed.

- virtual bool **operator<** (const double &d) const
Compares this object to another and returns true if this object is considered to be less than the one passed.
- std::string **toString** () const
- virtual double **doubleValue** () const
Answers the double value which the receiver represents.
- virtual float **floatValue** () const
Answers the float value which the receiver represents.
- virtual unsigned char **byteValue** () const
Answers the byte value which the receiver represents.
- virtual short **shortValue** () const
Answers the short value which the receiver represents.
- virtual int **intValue** () const
Answers the int value which the receiver represents.
- virtual long long **longValue** () const
Answers the long value which the receiver represents.
- bool **isInfinite** () const
- bool **isNaN** () const

Static Public Member Functions

- static int **compare** (double d1, double d2)
Compares the two specified double values.
- static long long **doubleToLongBits** (double value)
Returns a representation of the specified floating-point value according to the IEEE 754 floating-point "double format" bit layout.
- static long long **doubleToRawLongBits** (double value)
Returns a representation of the specified floating-point value according to the IEEE 754 floating-point "double format" bit layout, preserving Not-a-Number (NaN) values.
- static bool **isInfinite** (double value)
- static bool **isNaN** (double value)
- static double **longBitsToDouble** (long long bits)
Returns the double value corresponding to a given bit representation.
- static double **parseDouble** (const std::string value) throw (exceptions::NumberFormatException)
*Returns a new double initialized to the value represented by the specified string, as performed by the `valueOf` method of class **Double** (p. 1441).*

- static std::string **toHexString** (double value)
Returns a hexadecimal string representation of the double argument.
- static std::string **toString** (double value)
Returns a string representation of the double argument.
- static **Double** **valueOf** (double value)
*Returns a **Double** (p. 1441) instance representing the specified double value.*
- static **Double** **valueOf** (const std::string &value) throw (exceptions::NumberFormatException)
*Returns a **Double** (p. 1441) instance that wraps a primitive double which is parsed from the string value passed.*

Static Public Attributes

- static const int **SIZE** = 64
The size in bits of the primitive int type.
- static const double **MAX_VALUE**
The maximum value that the primitive type can hold.
- static const double **MIN_VALUE**
The minimum value that the primitive type can hold.
- static const double **NaN**
*Constant for the Not a **Number** (p. 2282) Value.*
- static const double **POSITIVE_INFINITY**
Constant for Positive Infinity.
- static const double **NEGATIVE_INFINITY**
Constant for Negative Infinitiy.

6.288.1 Constructor & Destructor Documentation

6.288.1.1 decaf::lang::Double::Double (double value)

Parameters

value - the primitive type to wrap

6.288.1.2 decaf::lang::Double::Double (const std::string & value) throw (exceptions::NumberFormatException)

Parameters

value - the string to convert to a primitive type to wrap

6.288.1.3 `virtual decaf::lang::Double::~~Double () [inline, virtual]`

6.288.2 Member Function Documentation

6.288.2.1 `virtual unsigned char decaf::lang::Double::byteValue () const [inline, virtual]`

Answers the byte value which the receiver represents.

Returns

byte the value of the receiver.

6.288.2.2 `static int decaf::lang::Double::compare (double d1, double d2) [static]`

Compares the two specified double values.

The sign of the integer value returned is the same as that of the integer that would be returned by the call: `new Double(d1).compareTo(new Double(d2))`

Parameters

d1 - the first double to compare

d2 - the second double to compare

Returns

the value 0 if *d1* is numerically equal to *d2*; a value less than 0 if *d1* is numerically less than *d2*; and a value greater than 0 if *d1* is numerically greater than *d2*.

6.288.2.3 `virtual int decaf::lang::Double::compareTo (const double & d) const [virtual]`

Compares this **Double** (p. 1441) instance with another.

Parameters

d - the **Double** (p. 1441) instance to be compared

Returns

zero if this object represents the same integer value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

Implements `decaf::lang::Comparable< double >` (p. 1010).

6.288.2.4 `virtual int decaf::lang::Double::compareTo (const Double & d) const [virtual]`

Compares this **Double** (p. 1441) instance with another.

Parameters

d - the **Double** (p. 1441) instance to be compared

Returns

zero if this object represents the same integer value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

6.288.2.5 static long long decaf::lang::Double::doubleToLongBits (double *value*) [static]

Returns a representation of the specified floating-point value according to the IEEE 754 floating-point "double format" bit layout.

Bit 63 (the bit that is selected by the mask 0x8000000000000000L) represents the sign of the floating-point number. Bits 62-52 (the bits that are selected by the mask 0x7ff0000000000000L) represent the exponent. Bits 51-0 (the bits that are selected by the mask 0x000fffffffffffffL) represent the significand (sometimes called the mantissa) of the floating-point number.

If the argument is positive infinity, the result is 0x7ff0000000000000L. If the argument is negative infinity, the result is 0xfff0000000000000L. If the argument is NaN, the result is 0x7ff8000000000000L.

In all cases, the result is a long integer that, when given to the longBitsToDouble(long) method, will produce a floating-point value the same as the argument to doubleToLongBits (except all NaN values are collapsed to a single "canonical" NaN value).

Parameters

value - double to be converted

Returns

the long long bits that make up the double

6.288.2.6 static long long decaf::lang::Double::doubleToRawLongBits (double *value*) [static]

Returns a representation of the specified floating-point value according to the IEEE 754 floating-point "double format" bit layout, preserving Not-a-Number (NaN) values.

Bit 63 (the bit that is selected by the mask 0x8000000000000000LL) represents the sign of the floating-point number. Bits 62-52 (the bits that are selected by the mask 0x7ff0000000000000L) represent the exponent. Bits 51-0 (the bits that are selected by the mask 0x000fffffffffffffL) represent the significand (sometimes called the mantissa) of the floating-point number.

If the argument is positive infinity, the result is 0x7ff0000000000000LL. If the argument is negative infinity, the result is 0xfff0000000000000LL. If the argument is NaN, the result is the long integer representing the actual NaN value. Unlike the doubleToLongBits method, doubleToRawLongBits does not collapse all the bit patterns encoding a NaN to a single "canonical" NaN value.

In all cases, the result is a long integer that, when given to the longBitsToDouble(long) method, will produce a floating-point value the same as the argument to doubleToRawLongBits.

Parameters

value - double to be converted

Returns

the long long bits that make up the double

6.288.2.7 `virtual double decaf::lang::Double::doubleValue () const [inline, virtual]`

Answers the double value which the receiver represents.

Returns

double the value of the receiver.

6.288.2.8 `bool decaf::lang::Double::equals (const Double & d) const [inline]`

Parameters

d - the **Double** (p. 1441) object to compare against.

Returns

true if the two **Double** (p. 1441) Objects have the same value.

6.288.2.9 `bool decaf::lang::Double::equals (const double & d) const [inline, virtual]`

Parameters

d - the **Double** (p. 1441) object to compare against.

Returns

true if the two **Double** (p. 1441) Objects have the same value.

Implements **decaf::lang::Comparable< double >** (p. 1010).

6.288.2.10 `virtual float decaf::lang::Double::floatValue () const [inline, virtual]`

Answers the float value which the receiver represents.

Returns

float the value of the receiver.

6.288.2.11 `virtual int decaf::lang::Double::intValue () const` [inline, virtual]

Answers the int value which the receiver represents.

Returns

int the value of the receiver.

6.288.2.12 `bool decaf::lang::Double::isInfinite () const`

Returns

true if the double is equal to positive infinity.

6.288.2.13 `static bool decaf::lang::Double::isInfinite (double value)` [static]

Parameters

value - The double to check.

Returns

true if the double is equal to infinity.

6.288.2.14 `bool decaf::lang::Double::isNaN () const`

Returns

true if the double is equal to NaN.

6.288.2.15 `static bool decaf::lang::Double::isNaN (double value)` [static]

Parameters

value - The double to check.

Returns

true if the double is equal to NaN.

6.288.2.16 `static double decaf::lang::Double::longBitsToDouble (long long bits)`
[static]

Returns the double value corresponding to a given bit representation.

The argument is considered to be a representation of a floating-point value according to the IEEE 754 floating-point "double format" bit layout.

If the argument is 0x7ff0000000000000L, the result is positive infinity. If the argument is 0xfff0000000000000L, the result is negative infinity. If the argument is any value in the range 0x7ff0000000000001L through 0x7fffffffffffffffL or in the range 0xfff0000000000001L through 0xfffffffffffffffL, the result is a NaN. No IEEE 754 floating-point operation provided by C++ can distinguish between two NaN values of the same type with different bit patterns. Distinct values of NaN are only distinguishable by use of the **Double.doubleToRawLongBits** (p. 1445) method.

Parameters

bits - the long long bits to convert to double

Returns

the double converted from the bits

6.288.2.17 `virtual long long decaf::lang::Double::longValue () const [inline, virtual]`

Answers the long value which the receiver represents.

Returns

long the value of the receiver.

6.288.2.18 `virtual bool decaf::lang::Double::operator< (const Double & d) const [inline, virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

Parameters

d - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

6.288.2.19 `virtual bool decaf::lang::Double::operator< (const double & d) const [inline, virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

Parameters

d - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

Implements `decaf::lang::Comparable< double >` (p.1011).

6.288.2.20 `virtual bool decaf::lang::Double::operator==(const Double & d)
const [inline, virtual]`

Compares equality between this object and the one passed.

Parameters

d - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

6.288.2.21 `virtual bool decaf::lang::Double::operator==(const double & d)
const [inline, virtual]`

Compares equality between this object and the one passed.

Parameters

d - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

Implements `decaf::lang::Comparable< double >` (p.1011).

6.288.2.22 `static double decaf::lang::Double::parseDouble (const std::string value
) throw (exceptions::NumberFormatException) [static]`

Returns a new double initialized to the value represented by the specified string, as performed by the `valueOf` method of class **Double** (p.1441).

Parameters

value - The string to parse to an double

Returns

a double parsed from the passed string

Exceptions

NumberFormatException

6.288.2.23 `virtual short decaf::lang::Double::shortValue () const [inline,
virtual]`

Answers the short value which the receiver represents.

Returns

short the value of the receiver.

6.288.2.24 `static std::string decaf::lang::Double::toHexString (double value)` `[static]`

Returns a hexadecimal string representation of the double argument.

All characters mentioned below are ASCII characters.

* If the argument is NaN, the result is the string "NaN". * Otherwise, the result is a string that represents the sign and magnitude (absolute value) of the argument. If the sign is negative, the first character of the result is '-'; if the sign is positive, no sign character appears in the result. As for the magnitude m: o If m is infinity, it is represented by the string "Infinity"; thus, positive infinity produces the result "Infinity" and negative infinity produces the result "-Infinity". o If m is zero, it is represented by the string "0x0.0p0"; thus, negative zero produces the result "-0x0.0p0" and positive zero produces the result "0x0.0p0". o If m is a double value with a normalized representation, substrings are used to represent the significand and exponent fields. The significand is represented by the characters "0x1." followed by a lowercase hexadecimal representation of the rest of the significand as a fraction. Trailing zeros in the hexadecimal representation are removed unless all the digits are zero, in which case a single zero is used. Next, the exponent is represented by "p" followed by a decimal string of the unbiased exponent as if produced by a call to **Integer.toString** (p.1664) on the exponent value. o If m is a double value with a subnormal representation, the significand is represented by the characters "0x0." followed by a hexadecimal representation of the rest of the significand as a fraction. Trailing zeros in the hexadecimal representation are removed. Next, the exponent is represented by "p-126". Note that there must be at least one nonzero digit in a subnormal significand.

Parameters

value - The double to convert to a string

Returns

the Hex formatted double string.

6.288.2.25 `std::string decaf::lang::Double::toString () const`

Returns

this **Double** (p.1441) Object as a String Representation

6.288.2.26 `static std::string decaf::lang::Double::toString (double value)` `[static]`

Returns a string representation of the double argument.

All characters mentioned below are ASCII characters.

If the argument is NaN, the result is the string "NaN". Otherwise, the result is a string that represents the sign and magnitude (absolute value) of the argument. If the sign is negative, the first character of the result is '-'; if the sign is positive, no sign character appears in the result. As for the magnitude m: o If m is infinity, it is represented by the characters "Infinity"; thus, positive infinity produces the result "Infinity" and negative infinity produces the result "-Infinity". o If m is zero, it is represented by the characters "0.0"; thus, negative zero produces the result "-0.0" and positive zero produces the result "0.0". o If m is greater than or equal to 10⁻³ but less than 10⁷, then it is represented as the integer part of m, in decimal form with no leading zeroes, followed by '.', followed by one or more decimal digits representing the fractional part of m. o If m

is less than 10^{-3} or greater than or equal to 10^7 , then it is represented in so-called "computerized scientific notation." Let n be the unique integer such that $10^n \leq m < 10^{n+1}$; then let a be the mathematically exact quotient of m and 10^n so that $1 \leq a < 10$. The magnitude is then represented as the integer part of a , as a single decimal digit, followed by '.', followed by decimal digits representing the fractional part of a , followed by the letter 'E', followed by a representation of n as a decimal integer, as produced by the method **Integer.toString(int)** (p. 1665).

Parameters

value - The double to convert to a string

Returns

the formatted double string.

6.288.2.27 static Double decaf::lang::Double::valueOf (double value) [static]

Returns a **Double** (p. 1441) instance representing the specified double value.

Parameters

value - double to wrap

Returns

new **Double** (p. 1441) instance wrapping the primitive value

6.288.2.28 static Double decaf::lang::Double::valueOf (const std::string & value) throw (exceptions::NumberFormatException) [static]

Returns a **Double** (p. 1441) instance that wraps a primitive double which is parsed from the string value passed.

Parameters

value - the string to parse

Returns

a new **Double** (p. 1441) instance wrapping the double parsed from value

Exceptions

NumberFormatException on error.

6.288.3 Field Documentation

6.288.3.1 const double decaf::lang::Double::MAX_VALUE [static]

The maximum value that the primitive type can hold.

6.288.3.2 const double decaf::lang::Double::MIN_VALUE [static]

The minimum value that the primitive type can hold.

6.288.3.3 `const double decaf::lang::Double::NaN` [static]

Constant for the Not a **N**umber (p. 2282) Value.

6.288.3.4 `const double decaf::lang::Double::NEGATIVE_INFINITY` [static]

Constant for Negative Infinitiy.

6.288.3.5 `const double decaf::lang::Double::POSITIVE_INFINITY` [static]

Constant for Positive Infinity.

6.288.3.6 `const int decaf::lang::Double::SIZE = 64` [static]

The size in bits of the primitive int type.

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Double.h`

6.289 `decaf::internal::nio::DoubleArrayBuffer` Class Reference

```
#include <src/main/decaf/internal/nio/DoubleArrayBuffer.h>
```

Inheritance diagram for `decaf::internal::nio::DoubleArrayBuffer`:

Public Member Functions

- **DoubleArrayBuffer** (`std::size_t` capacity, `bool` readOnly=false)
*Creates a **DoubleArrayBuffer** (p. 1452) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*
- **DoubleArrayBuffer** (`double *array`, `std::size_t` offset, `std::size_t` capacity, `bool` readOnly=false) throw (`decaf::lang::exceptions::NullPointerException`)
*Creates a **DoubleArrayBuffer** (p. 1452) object that wraps the given array.*
- **DoubleArrayBuffer** (**ByteArrayPerspective** &array, `std::size_t` offset, `std::size_t` length, `bool` readOnly=false) throw (`decaf::lang::exceptions::IndexOutOfBoundsException`)
*Creates a byte buffer that wraps the passed **ByteArrayPerspective** (p. 843) and start at the given offset.*
- **DoubleArrayBuffer** (`const DoubleArrayBuffer &other`)
*Create a **DoubleArrayBuffer** (p. 1452) that mirrors this one, meaning it shares a reference to this buffers **ByteArrayPerspective** (p. 843) and when changes are made to that data it is reflected in both.*

- virtual `~DoubleArrayBuffer ()`
- virtual `double * array () throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException)`
Returns the double array that backs this buffer (optional operation).
- virtual `std::size_t arrayOffset () throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException)`
Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).
- virtual `DoubleBuffer * asReadOnlyBuffer () const`
Creates a new, read-only double buffer that shares this buffer's content.
- virtual `DoubleBuffer & compact () throw (decaf::nio::ReadOnlyBufferException)`
Compacts this buffer.
- virtual `DoubleBuffer * duplicate ()`
Creates a new double buffer that shares this buffer's content.
- virtual `double get () throw (decaf::nio::BufferUnderflowException)`
Relative get method.
- virtual `double get (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)`
Absolute get method.
- virtual `bool hasArray () const`
Tells whether or not this buffer is backed by an accessible double array.
- virtual `bool isReadOnly () const`
Tells whether or not this buffer is read-only.
- virtual `DoubleBuffer & put (double value) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException)`
Writes the given doubles into this buffer at the current position, and then increments the position.
- virtual `DoubleBuffer & put (std::size_t index, double value) throw (lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException)`
Writes the given doubles into this buffer at the given index.
- virtual `DoubleBuffer * slice () const`
Creates a new DoubleBuffer whose content is a shared subsequence of this buffer's content.

Protected Member Functions

- virtual `void setReadOnly (bool value)`
*Sets this **ByteArrayBuffer** (p. 806) as Read-Only.*

6.289.1 Constructor & Destructor Documentation

6.289.1.1 `decaf::internal::nio::DoubleArrayBuffer::DoubleArrayBuffer (std::size_t capacity, bool readOnly = false)`

Creates a **DoubleArrayBuffer** (p.1452) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.

The array is initially created with all elements initialized to zero.

Parameters

capacity - size of the array, this is the limit we read and write to.

readOnly - should this buffer be read-only, default as false

6.289.1.2 `decaf::internal::nio::DoubleArrayBuffer::DoubleArrayBuffer (double * array, std::size_t offset, std::size_t capacity, bool readOnly = false) throw (decaf::lang::exceptions::NullPointerException)`

Creates a **DoubleArrayBuffer** (p.1452) object that wraps the given array.

If the own flag is set then it will delete this array when this object is deleted.

Parameters

array - array to wrap

offset - the position that is this buffers start pos.

capacity - size of the array, this is the limit we read and write to.

readOnly - should this buffer be read-only, default as false

Exceptions

NullPointerException if buffer is NULL

6.289.1.3 `decaf::internal::nio::DoubleArrayBuffer::DoubleArrayBuffer (ByteArrayPerspective & array, std::size_t offset, std::size_t length, bool readOnly = false) throw (decaf::lang::exceptions::IndexOutOfBoundsException)`

Creates a byte buffer that wraps the passed **ByteArrayPerspective** (p.843) and start at the given offset.

The capacity and limit of the new **DoubleArrayBuffer** (p.1452) will be that of the remaining capacity of the passed buffer.

Parameters

array - the **ByteArrayPerspective** (p.843) to wrap

offset - the offset into array where the buffer starts

length - the length of the array we are wrapping or limit.

readOnly - is this a readOnly buffer.

Exceptions

IndexOutOfBoundsException if offset is greater than array capacity.

6.289.1.4 decaf::internal::nio::DoubleArrayBuffer::DoubleArrayBuffer (const DoubleArrayBuffer & *other*)

Create a **DoubleArrayBuffer** (p. 1452) that mirrors this one, meaning it shares a reference to this buffers **ByteArrayPerspective** (p. 843) and when changes are made to that data it is reflected in both.

Parameters

other - the **DoubleArrayBuffer** (p. 1452) this one is to mirror.

6.289.1.5 virtual decaf::internal::nio::DoubleArrayBuffer::~~DoubleArrayBuffer () [virtual]

6.289.2 Member Function Documentation

6.289.2.1 virtual double* decaf::internal::nio::DoubleArrayBuffer::array () throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException) [virtual]

Returns the double array that backs this buffer (optional operation).

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

the array that backs this Buffer

Exceptions

ReadOnlyBufferException if this Buffer is read only.

UnsupportedOperationException if the underlying store has no array.

Implements **decaf::nio::DoubleBuffer** (p. 1462).

6.289.2.2 virtual std::size_t decaf::internal::nio::DoubleArrayBuffer::arrayOffset () throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException) [virtual]

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

The offset into the backing array where index zero starts.

Exceptions

ReadOnlyBufferException if this Buffer is read only.

UnsupportedOperationException if the underlying store has no array.

Implements **decaf::nio::DoubleBuffer** (p. 1463).

6.289.2.3 `virtual DoubleBuffer* decaf::internal::nio::DoubleArrayBuffer::asReadOnlyBuffer () const [virtual]`

Creates a new, read-only double buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns

The new, read-only double buffer which the caller then owns.

Implements **decaf::nio::DoubleBuffer** (p. 1463).

6.289.2.4 `virtual DoubleBuffer& decaf::internal::nio::DoubleArrayBuffer::compact () throw (decaf::nio::ReadOnlyBufferException) [virtual]`

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index `p = position()` (p. 746) is copied to index zero, the byte at index `p + 1` is copied to index one, and so forth until the byte at index `limit()` (p. 745) - 1 is copied to index `n = limit()` (p. 745) - 1 - `p`. The buffer's position is then set to `n+1` and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns

a reference to this **DoubleBuffer**

Exceptions

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::DoubleBuffer** (p. 1464).

6.289.2.5 `virtual DoubleBuffer* decaf::internal::nio::DoubleArrayBuffer::duplicate () [virtual]`

Creates a new double buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

a new double Buffer which the caller owns.

Implements **decaf::nio::DoubleBuffer** (p. 1464).

6.289.2.6 virtual double decaf::internal::nio::DoubleArrayBuffer::get () throw (decaf::nio::BufferUnderflowException) [virtual]

Relative get method.

Reads the value at this buffer's current position, and then increments the position.

Returns

the double at the current position

Exceptions

BufferUnderflowException if there no more data to return

Implements **decaf::nio::DoubleBuffer** (p. 1466).

6.289.2.7 virtual double decaf::internal::nio::DoubleArrayBuffer::get (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException) [virtual]

Absolute get method.

Reads the value at the given index.

Parameters

index - the index in the Buffer where the double is to be read

Returns

the double that is located at the given index

Exceptions

IndexOutOfBoundsException - If index is not smaller than the buffer's limit

Implements **decaf::nio::DoubleBuffer** (p. 1465).

6.289.2.8 virtual bool decaf::internal::nio::DoubleArrayBuffer::hasArray () const [inline, virtual]

Tells whether or not this buffer is backed by an accessible double array.

If this method returns true then the `array` and `arrayOffset` methods may safely be invoked. Sub-classes should override this method if they do not have a backing array as this class always returns true.

Returns

true if, and only if, this buffer is backed by an array and is not read-only

Implements **decaf::nio::DoubleBuffer** (p. 1466).

6.289.2.9 `virtual bool decaf::internal::nio::DoubleArrayBuffer::isReadOnly ()`
`const [inline, virtual]`

Tells whether or not this buffer is read-only.

Returns

true if, and only if, this buffer is read-only

6.289.2.10 `virtual DoubleBuffer& decaf::internal::nio::DoubleArrayBuffer::put`
`(std::size_t index, double value) throw`
`(lang::exceptions::IndexOutOfBoundsException,`
`decaf::nio::ReadOnlyBufferException) [virtual]`

Writes the given doubles into this buffer at the given index.

Parameters

index - position in the Buffer to write the data

value - the doubles to write.

Returns

a reference to this buffer

Exceptions

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::DoubleBuffer** (p. 1468).

6.289.2.11 `virtual DoubleBuffer& decaf::internal::nio::DoubleArrayBuffer::put`
`(double value) throw (decaf::nio::BufferOverflowException,`
`decaf::nio::ReadOnlyBufferException) [virtual]`

Writes the given doubles into this buffer at the current position, and then increments the position.

Parameters

value - the doubles value to be written

Returns

a reference to this buffer

Exceptions

BufferOverflowException - If this buffer's current position is not smaller than its limit

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::DoubleBuffer** (p. 1467).

6.289.2.12 `virtual void decaf::internal::nio::DoubleArrayBuffer::setReadOnly (bool value)` [inline, protected, virtual]

Sets this **ByteBuffer** (p. 806) as Read-Only.

Parameters

value - true if this buffer is to be read-only.

6.289.2.13 `virtual DoubleBuffer* decaf::internal::nio::DoubleArrayBuffer::slice () const` [virtual]

Creates a new **DoubleBuffer** whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the newly create **DoubleBuffer** which the caller owns.

Implements **decaf::nio::DoubleBuffer** (p. 1469).

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/nio/DoubleArrayBuffer.h`

6.290 decaf::nio::DoubleBuffer Class Reference

This class defines four categories of operations upon double buffers:

```
#include <src/main/decaf/nio/DoubleBuffer.h>
```

Inheritance diagram for **decaf::nio::DoubleBuffer**:

Public Member Functions

- virtual **~DoubleBuffer** ()
- virtual std::string **toString** () const
- virtual double * **array** ()=0 throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException)
Returns the double array that backs this buffer (optional operation).
- virtual std::size_t **arrayOffset** ()=0 throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException)
Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).
- virtual **DoubleBuffer** * **asReadOnlyBuffer** () const =0
Creates a new, read-only double buffer that shares this buffer's content.
- virtual **DoubleBuffer** & **compact** ()=0 throw (ReadOnlyBufferException)
Compacts this buffer.
- virtual **DoubleBuffer** * **duplicate** ()=0
Creates a new double buffer that shares this buffer's content.
- virtual double **get** ()=0 throw (BufferUnderflowException)
Relative get method.
- virtual double **get** (std::size_t index) const =0 throw (lang::exceptions::IndexOutOfBoundsException)
Absolute get method.
- **DoubleBuffer** & **get** (std::vector< double > buffer) throw (BufferUnderflowException)
Relative bulk get method.
- **DoubleBuffer** & **get** (double *buffer, std::size_t offset, std::size_t length) throw (BufferUnderflowException, lang::exceptions::NullPointerException)
Relative bulk get method.
- virtual bool **hasArray** () const =0
Tells whether or not this buffer is backed by an accessible double array.
- **DoubleBuffer** & **put** (**DoubleBuffer** &src) throw (BufferOverflowException, ReadOnlyBufferException, lang::exceptions::IllegalArgumentException)
This method transfers the doubles remaining in the given source buffer into this buffer.
- **DoubleBuffer** & **put** (const double *buffer, std::size_t offset, std::size_t length) throw (BufferOverflowException, ReadOnlyBufferException, lang::exceptions::NullPointerException)
This method transfers doubles into this buffer from the given source array.
- **DoubleBuffer** & **put** (std::vector< double > &buffer) throw (BufferOverflowException, ReadOnlyBufferException)
This method transfers the entire content of the given source doubles array into this buffer.

- virtual **DoubleBuffer** & **put** (double value)=0 throw (BufferOverflowException, ReadOnlyBufferException)
Writes the given doubles into this buffer at the current position, and then increments the position.
- virtual **DoubleBuffer** & **put** (std::size_t index, double value)=0 throw (lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException)
Writes the given doubles into this buffer at the given index.
- virtual **DoubleBuffer** * **slice** () const =0
*Creates a new **DoubleBuffer** (p. 1459) whose content is a shared subsequence of this buffer's content.*
- virtual int **compareTo** (const **DoubleBuffer** &value) const
Compares this object with the specified object for order.
- virtual bool **equals** (const **DoubleBuffer** &value) const
- virtual bool **operator==** (const **DoubleBuffer** &value) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const **DoubleBuffer** &value) const
Compares this object to another and returns true if this object is considered to be less than the one passed.

Static Public Member Functions

- static **DoubleBuffer** * **allocate** (std::size_t capacity)
Allocates a new Double buffer.
- static **DoubleBuffer** * **wrap** (double *array, std::size_t offset, std::size_t length) throw (lang::exceptions::NullPointerException)
*Wraps the passed buffer with a new **DoubleBuffer** (p. 1459).*
- static **DoubleBuffer** * **wrap** (std::vector< double > &buffer)
*Wraps the passed STL double Vector in a **DoubleBuffer** (p. 1459).*

Protected Member Functions

- **DoubleBuffer** (std::size_t capacity)
*Creates a **DoubleBuffer** (p. 1459) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*

6.290.1 Detailed Description

This class defines four categories of operations upon double buffers: o Absolute and relative get and put methods that read and write single doubles; o Relative bulk get methods that transfer

contiguous sequences of doubles from this buffer into an array; and o Relative bulk put methods that transfer contiguous sequences of doubles from a double array or some other double buffer into this buffer o Methods for compacting, duplicating, and slicing a double buffer.

Double buffers can be created either by allocation, which allocates space for the buffer's content, by wrapping an existing double array into a buffer, or by creating a view of an existing byte buffer

Methods in this class that do not otherwise have a value to return are specified to return the buffer upon which they are invoked. This allows method invocations to be chained.

6.290.2 Constructor & Destructor Documentation

6.290.2.1 `decaf::nio::DoubleBuffer::DoubleBuffer (std::size_t capacity)` [protected]

Creates a **DoubleBuffer** (p.1459) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.

The array is initially created with all elements initialized to zero.

Parameters

capacity - size and limit of the **Buffer** (p.741) in doubles

6.290.2.2 `virtual decaf::nio::DoubleBuffer::~~DoubleBuffer ()` [inline, virtual]

6.290.3 Member Function Documentation

6.290.3.1 `static DoubleBuffer* decaf::nio::DoubleBuffer::allocate (std::size_t capacity)` [static]

Allocates a new Double buffer.

The new buffer's position will be zero, its limit will be its capacity, and its mark will be undefined. It will have a backing array, and its array offset will be zero.

Parameters

capacity - The size of the Double buffer in doubles

Returns

the **DoubleBuffer** (p.1459) that was allocated, caller owns.

6.290.3.2 `virtual double* decaf::nio::DoubleBuffer::array ()` throw (`decaf::lang::exceptions::UnsupportedOperationException`, `ReadOnlyBufferException`) [pure virtual]

Returns the double array that backs this buffer (optional operation).

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

the array that backs this **Buffer** (p. 741)

Exceptions

ReadOnlyBufferException (p. 2520) if this **Buffer** (p. 741) is read only.

UnsupportedOperationException if the underlying store has no array.

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 1455).

6.290.3.3 virtual std::size_t decaf::nio::DoubleBuffer::arrayOffset ()
throw (decaf::lang::exceptions::UnsupportedOperationException,
ReadOnlyBufferException) [pure virtual]

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

The offset into the backing array where index zero starts.

Exceptions

ReadOnlyBufferException (p. 2520) if this **Buffer** (p. 741) is read only.

UnsupportedOperationException if the underlying store has no array.

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 1455).

6.290.3.4 virtual DoubleBuffer* decaf::nio::DoubleBuffer::asReadOnlyBuffer ()
const [pure virtual]

Creates a new, read-only double buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the `duplicate` method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns

The new, read-only double buffer which the caller then owns.

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 1456).

6.290.3.5 **virtual DoubleBuffer& decaf::nio::DoubleBuffer::compact () throw (ReadOnlyBufferException)** [pure virtual]

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index $p = \text{position}()$ (p. 746) is copied to index zero, the byte at index $p + 1$ is copied to index one, and so forth until the byte at index $\text{limit}()$ (p. 745) - 1 is copied to index $n = \text{limit}()$ (p. 745) - 1 - p . The buffer's position is then set to $n+1$ and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns

a reference to this **DoubleBuffer** (p. 1459)

Exceptions

ReadOnlyBufferException (p. 2520) - If this buffer is read-only

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 1456).

6.290.3.6 **virtual int decaf::nio::DoubleBuffer::compareTo (const DoubleBuffer & value) const** [virtual]

Compares this object with the specified object for order.

Returns a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

Parameters

value - the Object to be compared.

Returns

a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

6.290.3.7 **virtual DoubleBuffer* decaf::nio::DoubleBuffer::duplicate ()** [pure virtual]

Creates a new double buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

a new double **Buffer** (p. 741) which the caller owns.

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 1456).

6.290.3.8 virtual bool decaf::nio::DoubleBuffer::equals (const DoubleBuffer & *value*) const [virtual]

Returns

true if this value is considered equal to the passed value.

6.290.3.9 DoubleBuffer& decaf::nio::DoubleBuffer::get (std::vector< double > *buffer*) throw (BufferUnderflowException)

Relative bulk get method.

This method transfers values from this buffer into the given destination vector. An invocation of this method of the form `src.get(a)` behaves in exactly the same way as the invocation. The vector must be sized to the amount of data that is to be read, that is to say, the caller should call `buffer.resize(N)` before calling this get method.

Returns

a reference to this **Buffer** (p. 741)

Exceptions

BufferUnderflowException (p. 774) - If there are fewer than length doubles remaining in this buffer

6.290.3.10 virtual double decaf::nio::DoubleBuffer::get (std::size_t *index*) const throw (lang::exceptions::IndexOutOfBoundsException) [pure virtual]

Absolute get method.

Reads the value at the given index.

Parameters

index - the index in the **Buffer** (p. 741) where the double is to be read

Returns

the double that is located at the given index

Exceptions

IndexOutOfBoundsException - If index is not smaller than the buffer's limit

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 1457).

6.290.3.11 DoubleBuffer& decaf::nio::DoubleBuffer::get (double * *buffer*, std::size_t *offset*, std::size_t *length*) throw (BufferUnderflowException, lang::exceptions::NullPointerException)

Relative bulk get method.

This method transfers doubles from this buffer into the given destination array. If there are fewer doubles remaining in the buffer than are required to satisfy the request, that is, if `length > remaining()` (p. 746), then no bytes are transferred and a **BufferUnderflowException** (p. 774) is thrown.

Otherwise, this method copies `length` doubles from this buffer into the given array, starting at the current position of this buffer and at the given offset in the array. The position of this buffer is then incremented by `length`.

Parameters

buffer - pointer to an allocated buffer to fill
offset - position in the buffer to start filling
length - amount of data to put in the passed buffer

Returns

a reference to this **Buffer** (p. 741)

Exceptions

BufferUnderflowException (p. 774) - If there are fewer than `length` doubles remaining in this buffer
NullPointerException if the passed buffer is null.

6.290.3.12 `virtual double decaf::nio::DoubleBuffer::get () throw (BufferUnderflowException) [pure virtual]`

Relative get method.

Reads the value at this buffer's current position, and then increments the position.

Returns

the double at the current position

Exceptions

BufferUnderflowException (p. 774) if there no more data to return

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 1457).

6.290.3.13 `virtual bool decaf::nio::DoubleBuffer::hasArray () const [pure virtual]`

Tells whether or not this buffer is backed by an accessible double array.

If this method returns true then the `array` and `arrayOffset` methods may safely be invoked. Sub-classes should override this method if they do not have a backing array as this class always returns true.

Returns

true if, and only if, this buffer is backed by an array and is not read-only

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 1457).

6.290.3.14 `virtual bool decaf::nio::DoubleBuffer::operator< (const DoubleBuffer & value) const [virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

Parameters

value - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

6.290.3.15 `virtual bool decaf::nio::DoubleBuffer::operator== (const DoubleBuffer & value) const [virtual]`

Compares equality between this object and the one passed.

Parameters

value - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

6.290.3.16 `virtual DoubleBuffer& decaf::nio::DoubleBuffer::put (double value) throw (BufferOverflowException, ReadOnlyBufferException) [pure virtual]`

Writes the given doubles into this buffer at the current position, and then increments the position.

Parameters

value - the doubles value to be written

Returns

a reference to this buffer

Exceptions

BufferOverflowException (p. 771) - If this buffer's current position is not smaller than its limit

ReadOnlyBufferException (p. 2520) - If this buffer is read-only

Implemented in `decaf::internal::nio::DoubleArrayBuffer` (p. 1458).

6.290.3.17 `virtual DoubleBuffer& decaf::nio::DoubleBuffer::put (std::size_t index, double value) throw (lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException) [pure virtual]`

Writes the given doubles into this buffer at the given index.

Parameters

index - position in the **Buffer** (p. 741) to write the data
value - the doubles to write.

Returns

a reference to this buffer

Exceptions

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException (p. 2520) - If this buffer is read-only

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 1458).

6.290.3.18 `DoubleBuffer& decaf::nio::DoubleBuffer::put (const double * buffer, std::size_t offset, std::size_t length) throw (BufferOverflowException, ReadOnlyBufferException, lang::exceptions::NullPointerException)`

This method transfers doubles into this buffer from the given source array.

If there are more doubles to be copied from the array than remain in this buffer, that is, if `length > remaining()` (p. 746), then no doubles are transferred and a **BufferOverflowException** (p. 771) is thrown.

Otherwise, this method copies `length` bytes from the given array into this buffer, starting at the given offset in the array and at the current position of this buffer. The position of this buffer is then incremented by `length`.

Parameters

buffer- The array from which doubles are to be read
offset- The offset within the array of the first double to be read;
length - The number of doubles to be read from the given array

Returns

a reference to this buffer

Exceptions

BufferOverflowException (p. 771) - If there is insufficient space in this buffer

ReadOnlyBufferException (p. 2520) - If this buffer is read-only

NullPointerException if the passed buffer is null.

6.290.3.19 DoubleBuffer& decaf::nio::DoubleBuffer::put (std::vector< double > & buffer) throw (BufferOverflowException, ReadOnlyBufferException)

This method transfers the entire content of the given source doubles array into this buffer.

This is the same as calling put(&buffer[0], 0, buffer.size()

Parameters

buffer - The buffer whose contents are copied to this **DoubleBuffer** (p.1459)

Returns

a reference to this buffer

Exceptions

BufferOverflowException (p. 771) - If there is insufficient space in this buffer

ReadOnlyBufferException (p. 2520) - If this buffer is read-only

6.290.3.20 DoubleBuffer& decaf::nio::DoubleBuffer::put (DoubleBuffer & src) throw (BufferOverflowException, ReadOnlyBufferException, lang::exceptions::IllegalArgumentException)

This method transfers the doubles remaining in the given source buffer into this buffer.

If there are more doubles remaining in the source buffer than in this buffer, that is, if src.remaining() > remaining() (p.746), then no doubles are transferred and a **BufferOverflowException** (p. 771) is thrown.

Otherwise, this method copies n = src.remaining() doubles from the given buffer into this buffer, starting at each buffer's current position. The positions of both buffers are then incremented by n.

Parameters

src - the buffer to take doubles from an place in this one.

Returns

a reference to this buffer

Exceptions

BufferOverflowException (p. 771) - If there is insufficient space in this buffer for the remaining doubles in the source buffer

IllegalArgumentException - If the source buffer is this buffer

ReadOnlyBufferException (p. 2520) - If this buffer is read-only

6.290.3.21 virtual DoubleBuffer* decaf::nio::DoubleBuffer::slice () const [pure virtual]

Creates a new **DoubleBuffer** (p.1459) whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the newly create **DoubleBuffer** (p. 1459) which the caller owns.

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 1459).

6.290.3.22 `virtual std::string decaf::nio::DoubleBuffer::toString () const`
[virtual]

Returns

a `std::string` describing this object

6.290.3.23 `static DoubleBuffer* decaf::nio::DoubleBuffer::wrap (double
* array, std::size_t offset, std::size_t length) throw (`
`lang::exceptions::NullPointerException)` [static]

Wraps the passed buffer with a new **DoubleBuffer** (p. 1459).

The new buffer will be backed by the given double array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be `array.length`, its position will be `offset`, its limit will be `offset + length`, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

array - The array that will back the new buffer

offset - The offset of the subarray to be used

length - The length of the subarray to be used

Returns

a new **DoubleBuffer** (p. 1459) that is backed by buffer, caller owns.

6.290.3.24 `static DoubleBuffer* decaf::nio::DoubleBuffer::wrap (std::vector<`
`double > & buffer)` [static]

Wraps the passed STL double Vector in a **DoubleBuffer** (p. 1459).

The new buffer will be backed by the given double array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be `buffer.size()`, its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

buffer - The vector that will back the new buffer, the vector must have been sized to the desired size already by calling `vector.resize(N)`.

Returns

- a new **DoubleBuffer** (p. 1459) that is backed by buffer, caller owns.

The documentation for this class was generated from the following file:

- src/main/decaf/nio/**DoubleBuffer.h**

6.291 decaf::lang::DYNAMIC_CAST_TOKEN Struct Reference

```
#include <src/main/decaf/lang/Pointer.h>
```

The documentation for this struct was generated from the following file:

- src/main/decaf/lang/**Pointer.h**

6.292 activemq::cmsutil::DynamicDestinationResolver Class Reference

Resolves a CMS destination name to a **Destination**.

```
#include <src/main/activemq/cmsutil/DynamicDestinationResolver.h>
```

Inheritance diagram for `activemq::cmsutil::DynamicDestinationResolver`:

Data Structures

- class **SessionResolver**

Manages maps of names to topics and queues for a single session.

Public Member Functions

- virtual **~DynamicDestinationResolver** ()
- virtual void **init** (**ResourceLifecycleManager** *mgr)

Initializes this destination resolver for use.

- virtual void **destroy** ()

Destroys any allocated resources.

- virtual **cms::Destination** * **resolveDestinationName** (**cms::Session** *session, const std::string &destName, bool pubSubDomain) throw (cms::CMSException)

Resolves the given name to a destination.

6.292.1 Detailed Description

Resolves a CMS destination name to a `Destination`.

6.292.2 Constructor & Destructor Documentation

6.292.2.1 `virtual`
`activemq::cmsutil::DynamicDestinationResolver::~DynamicDestinationResolver`
`()` [virtual]

6.292.3 Member Function Documentation

6.292.3.1 `virtual void` `activemq::cmsutil::DynamicDestinationResolver::destroy` (
`)` [virtual]

Destroys any allocated resources.

Implements `activemq::cmsutil::DestinationResolver` (p. 1415).

6.292.3.2 `virtual void` `activemq::cmsutil::DynamicDestinationResolver::init` (
`ResourceLifecycleManager * mgr`) [inline, virtual]

Initializes this destination resolver for use.

Ensures that any previously allocated resources are first destroyed (e.g. calls `destroy()` (p. 1472)).

Parameters

mgr the resource lifecycle manager.

Implements `activemq::cmsutil::DestinationResolver` (p. 1416).

6.292.3.3 `virtual cms::Destination*` `activemq::cmsutil::DynamicDestinationResolver::resolveDestinationName`
`(cms::Session * session, const std::string & destName, bool`
`pubSubDomain) throw (cms::CMSException)` [virtual]

Resolves the given name to a destination.

If `pubSubDomain` is true, a topic will be returned, otherwise a queue will be returned.

Parameters

session the session for which to retrieve resolve the destination.

destName the name to be resolved.

pubSubDomain If true, the name will be resolved to a Topic, otherwise a Queue.

Returns

the resolved destination

Exceptions

cms::CMSException (p. 960) if resolution failed.

Implements **activemq::cmsutil::DestinationResolver** (p. 1416).

The documentation for this class was generated from the following file:

- `src/main/activemq/cmsutil/DynamicDestinationResolver.h`

6.293 decaf::util::Map< K, V, COMPARATOR >::Entry Class Reference

```
#include <src/main/decaf/util/Map.h>
```

Public Member Functions

- **Entry** ()
- virtual **~Entry** ()
- virtual const K & **getKey** () const =0
- virtual const V & **getValue** () const =0
- virtual void **setValue** (const V &value)=0

```
template<typename K, typename V, typename COMPARATOR = std::less<K>>
class decaf::util::Map< K, V, COMPARATOR >::Entry
```

6.293.1 Constructor & Destructor Documentation

6.293.1.1 `template<typename K, typename V, typename COMPARATOR = std::less<K>> decaf::util::Map< K, V, COMPARATOR >::Entry::Entry () [inline]`

6.293.1.2 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual decaf::util::Map< K, V, COMPARATOR >::Entry::~Entry () [inline, virtual]`

6.293.2 Member Function Documentation

6.293.2.1 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual const K& decaf::util::Map< K, V, COMPARATOR >::Entry::getKey () const [pure virtual]`

6.293.2.2 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual const V& decaf::util::Map< K, V, COMPARATOR >::Entry::getValue () const [pure virtual]`

6.293.2.3 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::Map< K, V, COMPARATOR >::Entry::setValue (const V & value) [pure virtual]`

The documentation for this class was generated from the following file:

- `src/main/decaf/util/Map.h`

6.294 decaf::io::EOFException Class Reference

```
#include <src/main/decaf/io/EOFException.h>
```

Inheritance diagram for decaf::io::EOFException:

Public Member Functions

- **EOFException** () throw ()
Default Constructor.
- **EOFException** (const lang::Exception &ex) throw ()
Copy Constructor.
- **EOFException** (const EOFException &ex) throw ()
Copy Constructor.
- **EOFException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **EOFException** (const std::exception *cause) throw ()
Constructor.
- **EOFException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor.
- virtual **EOFException** * clone () const
Clones this exception.
- virtual ~**EOFException** () throw ()

6.294.1 Constructor & Destructor Documentation

6.294.1.1 decaf::io::EOFException::EOFException () throw () [inline]

Default Constructor.

6.294.1.2 decaf::io::EOFException::EOFException (const lang::Exception & ex) throw () [inline]

Copy Constructor.

Parameters

ex the exception to copy

6.294.1.3 decaf::io::EOFException::EOFException (const EOFException & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex the exception to copy, which is an instance of this type

6.294.1.4 decaf::io::EOFException::EOFException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.294.1.5 decaf::io::EOFException::EOFException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.294.1.6 decaf::io::EOFException::EOFException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor.

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.294.1.7 `virtual decaf::io::EOFException::~~EOFException () throw () [inline, virtual]`

6.294.2 Member Function Documentation

6.294.2.1 `virtual EOFException* decaf::io::EOFException::clone () const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

a new instance of an Exception that is a copy of this one.

Reimplemented from `decaf::io::IOException` (p. 1709).

The documentation for this class was generated from the following file:

- `src/main/decaf/io/EOFException.h`

6.295 decaf::lang::Exception Class Reference

```
#include <src/main/decaf/lang/Exception.h>
```

Inheritance diagram for `decaf::lang::Exception`:

Public Member Functions

- **Exception** () throw ()
Default Constructor.
- **Exception** (const **Exception** &ex) throw ()
Copy Constructor.
- **Exception** (const std::exception *cause) throw ()
Constructor.
- **Exception** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **Exception** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **~Exception** () throw ()
- virtual std::string **getMessage** () const
Gets the message for this exception.

- virtual const std::exception * **getCause** () const
Gets the exception that caused this one to be thrown, this allows for chaining of exceptions in the case of a method that throws only a particular exception but wishes to allow for the real causal exception to be passed only in case the caller knows about that type of exception and wishes to respond to it.
- virtual void **initCause** (const std::exception *cause)
Initializes the contained cause exception with the one given.
- virtual const char * **what** () const throw ()
Implement method from std::exception.
- virtual void **setMessage** (const char *msg,...)
Sets the cause for this exception.
- virtual void **setMark** (const char *file, const int lineNumber)
Adds a file/line number to the stack trace.
- virtual **Exception** * **clone** () const
Clones this exception.
- virtual std::vector< std::pair< std::string, int > > **getStackTrace** () const
Provides the stack trace for every point where this exception was caught, marked, and rethrown.
- virtual void **printStackTrace** () const
Prints the stack trace to std::err.
- virtual void **printStackTrace** (std::ostream &stream) const
Prints the stack trace to the given output stream.
- virtual std::string **getStackTraceString** () const
Gets the stack trace as one contiguous string.
- virtual **Exception** & **operator=** (const **Exception** &ex)
Assignment operator.

Protected Member Functions

- virtual void **setStackTrace** (const std::vector< std::pair< std::string, int > > &trace)
- virtual void **buildMessage** (const char *format, va_list &args)

Protected Attributes

- std::string **message**
The cause of this exception.
- std::exception * **cause**

The **Exception** (p. 1476) that caused this one to be thrown.

- `std::vector< std::pair< std::string, int > > stackTrace`

The stack trace.

6.295.1 Constructor & Destructor Documentation

6.295.1.1 `decaf::lang::Exception::Exception () throw ()`

Default Constructor.

Referenced by `decaf::util::concurrent::BrokenBarrierException::BrokenBarrierException()`,
`decaf::util::concurrent::CancellationException::CancellationException()`,
`decaf::util::concurrent::ExecutionException::ExecutionException()`,
`cafe::net::HttpRetryException::HttpRetryException()`, `decaf::net::MalformedURLException::MalformedURLException()`,
`decaf::net::ProtocolException::ProtocolException()`, `decaf::util::concurrent::RejectedExecutionException::RejectedExecutionException()`,
`decaf::net::SocketTimeoutException::SocketTimeoutException()`,
`cafe::util::concurrent::TimeoutException::TimeoutException()`,
`cafe::net::UnknownHostException::UnknownHostException()`, and
`cafe::net::UnknownServiceException::UnknownServiceException()`.

6.295.1.2 `decaf::lang::Exception::Exception (const Exception & ex) throw ()`

Copy Constructor.

Parameters

ex The **Exception** (p. 1476) instance to copy.

6.295.1.3 `decaf::lang::Exception::Exception (const std::exception * cause) throw ()`

Constructor.

Parameters

cause Pointer (p. 2343) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.295.1.4 `decaf::lang::Exception::Exception (const char * file, const int lineNumber, const char * msg, ...) throw ()`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.295.1.5 `decaf::lang::Exception::Exception (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw ()`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.295.1.6 `virtual decaf::lang::Exception::~~Exception () throw () [virtual]`

6.295.2 Member Function Documentation

6.295.2.1 `virtual void decaf::lang::Exception::buildMessage (const char * format, va_list & vargs) [protected, virtual]`

Referenced by `decaf::lang::exceptions::NumberFormatException::NumberFormatException()`.

6.295.2.2 `virtual Exception* decaf::lang::Exception::clone () const [virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

Copy of this **Exception** (p.1476) object

Implements **decaf::lang::Throwable** (p.2984).

Reimplemented in **activemq::exceptions::ActiveMQException** (p.271), **activemq::exceptions::BrokerException** (p.691), **decaf::io::EOFException** (p.1476), **decaf::io::InterruptedIOException** (p.1695), **decaf::io::IOException** (p.1709), **decaf::io::UnsupportedEncodingException** (p.3092), **decaf::io::UTFDataFormatException** (p.3141), **decaf::lang::exceptions::ClassCastException** (p.950), **decaf::lang::exceptions::IllegalArgumentException** (p.1615), **decaf::lang::exceptions::IllegalMonitorStateException** (p.1618), **decaf::lang::exceptions::IllegalStateException** (p.1620), **decaf::lang::exceptions::IllegalThreadStateException** (p.1624), **decaf::lang::exceptions::IndexOutOfBoundsException** (p.1629), **decaf::lang::exceptions::InterruptedException** (p.1693), **decaf::lang::exceptions::InvalidStateException** (p.1706), **decaf::lang::exceptions::NoSuchElementException** (p.2276), **decaf::lang::exceptions::NullPointerException** (p.2281), **decaf::lang::exceptions::NumberFormatException** (p.2286),

decaf::lang::exceptions::RuntimeException (p. 2646), **decaf::lang::exceptions::UnsupportedOperationException** (p. 3096),
decaf::net::BindException (p. 665), **decaf::net::ConnectException** (p. 1051), **decaf::net::HttpRetryException** (p. 1613), **decaf::net::MalformedURLException** (p. 1970),
decaf::net::NoRouteToHostException (p. 2271), **decaf::net::PortUnreachableException** (p. 2370), **decaf::net::ProtocolException** (p. 2506), **decaf::net::SocketException** (p. 2794),
decaf::net::SocketTimeoutException (p. 2811), **decaf::net::UnknownHostException** (p. 3087), **decaf::net::UnknownServiceException** (p. 3089),
decaf::net::URISyntaxException (p. 3125), **decaf::nio::BufferOverflowException** (p. 773), **decaf::nio::BufferUnderflowException** (p. 776),
decaf::nio::InvalidMarkException (p. 1703), **decaf::nio::ReadOnlyBufferException** (p. 2522), **decaf::util::concurrent::BrokenBarrierException** (p. 685),
decaf::util::concurrent::CancellationException (p. 900), **decaf::util::concurrent::ExecutionException** (p. 1509),
decaf::util::concurrent::RejectedExecutionException (p. 2535), and **decaf::util::concurrent::TimeoutException** (p. 2989).

6.295.2.3 `virtual const std::exception* decaf::lang::Exception::getCause () const`
`[inline, virtual]`

Gets the exception that caused this one to be thrown, this allows for chaining of exceptions in the case of a method that throws only a particular exception but wishes to allow for the real causal exception to be passed only in case the caller knows about that type of exception and wishes to respond to it.

Returns

a const pointer reference to the causal exception, if there was no cause associated with this exception then NULL is returned.

Implements **decaf::lang::Throwable** (p. 2985).

6.295.2.4 `virtual std::string decaf::lang::Exception::getMessage () const`
`[inline, virtual]`

Gets the message for this exception.

Returns

Text formatted error message

Implements **decaf::lang::Throwable** (p. 2985).

6.295.2.5 `virtual std::vector< std::pair< std::string, int> > decaf::lang::Exception::getStackTrace () const` `[virtual]`

Provides the stack trace for every point where this exception was caught, marked, and rethrown.

The first item in the returned vector is the first point where the mark was set (e.g. where the exception was created).

Returns

the stack trace.

Implements **decaf::lang::Throwable** (p. 2985).

6.295.2.6 `virtual std::string decaf::lang::Exception::getStackTraceString () const [virtual]`

Gets the stack trace as one contiguous string.

Returns

string with formatted stack trace data

Implements **decaf::lang::Throwable** (p. 2986).

6.295.2.7 `virtual void decaf::lang::Exception::initCause (const std::exception * cause) [virtual]`

Initializes the contained cause exception with the one given.

A copy is made to avoid ownership issues.

Parameters

cause The exception that was the cause of this one.

Implements **decaf::lang::Throwable** (p. 2986).

6.295.2.8 `virtual Exception& decaf::lang::Exception::operator= (const Exception & ex) [virtual]`

Assignment operator.

Parameters

ex const reference to another **Exception** (p. 1476)

6.295.2.9 `virtual void decaf::lang::Exception::printStackTrace () const [virtual]`

Prints the stack trace to std::err.

Implements **decaf::lang::Throwable** (p. 2986).

6.295.2.10 `virtual void decaf::lang::Exception::printStackTrace (std::ostream & stream) const [virtual]`

Prints the stack trace to the given output stream.

Parameters

stream the target output stream.

Implements **decaf::lang::Throwable** (p. 2986).

6.295.2.11 `virtual void decaf::lang::Exception::setMark (const char * file, const int lineNumber)` [virtual]

Adds a file/line number to the stack trace.

Parameters

file The name of the file calling this method (use `__FILE__`).

lineNumber The line number in the calling file (use `__LINE__`).

Implements **decaf::lang::Throwable** (p. 2986).

Referenced by `activemq::exceptions::BrokerException::BrokerException()`, and `decaf::lang::exceptions::NumberFormatException::NumberFormatException()`.

6.295.2.12 `virtual void decaf::lang::Exception::setMessage (const char * msg, ...)` [virtual]

Sets the cause for this exception.

Parameters

msg the format string for the msg.

... params to format into the string

Referenced by `activemq::exceptions::BrokerException::BrokerException()`.

6.295.2.13 `virtual void decaf::lang::Exception::setStackTrace (const std::vector< std::pair< std::string, int > > & trace)` [protected, virtual]

6.295.2.14 `virtual const char* decaf::lang::Exception::what () const throw ()` [inline, virtual]

Implement method from `std::exception`.

Returns

the `const char*` of `getMessage()` (p. 1480).

6.295.3 Field Documentation

6.295.3.1 `std::exception* decaf::lang::Exception::cause` [protected]

The **Exception** (p. 1476) that caused this one to be thrown.

6.295.3.2 `std::string decaf::lang::Exception::message` [protected]

The cause of this exception.

6.295.3.3 std::vector< std::pair< std::string, int> > decaf::lang::Exception::stackTrace [protected]

The stack trace.

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**Exception.h**

6.296 cms::ExceptionListener Class Reference

If a CMS provider detects a serious problem, it notifies the client application through an `ExceptionListener` (p. 1483) that is registered with the `Connection` (p. 1052).

```
#include <src/main/cms/ExceptionListener.h>
```

Public Member Functions

- virtual `~ExceptionListener ()`
- virtual void `onException (const cms::CMSException &ex)=0`
Called when an exception occurs.

6.296.1 Detailed Description

If a CMS provider detects a serious problem, it notifies the client application through an `ExceptionListener` (p. 1483) that is registered with the `Connection` (p. 1052). An exception listener allows a client to be notified of a problem asynchronously. Some connections only consume messages via the asynchronous event mechanism so they would have no other way to learn that their connection has failed.

Since

1.0

6.296.2 Constructor & Destructor Documentation

- 6.296.2.1** virtual `cms::ExceptionListener::~ExceptionListener ()` [inline, virtual]

6.296.3 Member Function Documentation

- 6.296.3.1** virtual void `cms::ExceptionListener::onException (const cms::CMSException & ex)` [pure virtual]

Called when an exception occurs.

Once notified of an exception the caller should no longer use the resource that generated the exception.

Parameters

ex Exception Object that occurred.

The documentation for this class was generated from the following file:

- `src/main/cms/ExceptionListener.h`

6.297 activemq::commands::ExceptionResponse Class Reference

```
#include <src/main/activemq/commands/ExceptionResponse.h>
```

Inheritance diagram for `activemq::commands::ExceptionResponse`:

Public Member Functions

- **ExceptionResponse** ()
- virtual **~ExceptionResponse** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **ExceptionResponse * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1372) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **BrokerError** > & **getException** () const
- virtual **Pointer**< **BrokerError** > & **getException** ()
- virtual void **setException** (const **Pointer**< **BrokerError** > &exception)

Static Public Attributes

- static const unsigned char **ID_EXCEPTIONRESPONSE** = 31

Protected Member Functions

- **ExceptionResponse** (const **ExceptionResponse** &)
- **ExceptionResponse** & **operator=** (const **ExceptionResponse** &)

Protected Attributes

- `Pointer< BrokerError > exception`

6.297.1 Constructor & Destructor Documentation

6.297.1.1 `activemq::commands::ExceptionResponse::ExceptionResponse (const ExceptionResponse &) [inline, protected]`

6.297.1.2 `activemq::commands::ExceptionResponse::ExceptionResponse ()`

6.297.1.3 `virtual activemq::commands::ExceptionResponse::~~ExceptionResponse () [virtual]`

6.297.2 Member Function Documentation

6.297.2.1 `virtual ExceptionResponse* activemq::commands::ExceptionResponse::cloneDataStructure () const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Reimplemented from `activemq::commands::Response` (p.2612).

6.297.2.2 `virtual void activemq::commands::ExceptionResponse::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

Reimplemented from `activemq::commands::Response` (p.2612).

6.297.2.3 `virtual bool activemq::commands::ExceptionResponse::equals (const DataStructure * value) const [virtual]`

Compares the `DataStructure` (p.1372) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::Response` (p.2613).

6.297.2.4 `virtual unsigned char activemq::commands::ExceptionResponse::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataStructure** (p. 1372) type copy.

Reimplemented from **activemq::commands::Response** (p. 2613).

6.297.2.5 `virtual Pointer<BrokerError>& activemq::commands::ExceptionResponse::getException () [virtual]`

6.297.2.6 `virtual const Pointer<BrokerError>& activemq::commands::ExceptionResponse::getException () const [virtual]`

6.297.2.7 `ExceptionResponse& activemq::commands::ExceptionResponse::operator= (const ExceptionResponse &) [inline, protected]`

6.297.2.8 `virtual void activemq::commands::ExceptionResponse::setException (const Pointer< BrokerError > & exception) [virtual]`

6.297.2.9 `virtual std::string activemq::commands::ExceptionResponse::toString () const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1372) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::Response** (p. 2614).

6.297.3 Field Documentation

6.297.3.1 `Pointer<BrokerError> activemq::commands::ExceptionResponse::exception [protected]`

6.297.3.2 `const unsigned char activemq::commands::ExceptionResponse::ID _ - EXCEPTIONRESPONSE = 31 [static]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ExceptionResponse.h`

6.298 activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1487).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ExceptionResponseMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller:

Public Member Functions

- **ExceptionResponseMarshaller** ()
- virtual **~ExceptionResponseMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.298.1 Detailed Description

Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p.1487). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.298.2 Constructor & Destructor Documentation

6.298.2.1 `activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller::ExceptionRe`
`() [inline]`

6.298.2.2 `virtual`
`activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller::~~ExceptionR`
`() [inline, virtual]`

6.298.3 Member Function Documentation

6.298.3.1 `virtual commands::DataStructure* ac-`
`tivemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller::createObject`
`() const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 2621).

6.298.3.2 `virtual unsigned char ac-`
`tivemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller::getDataStructu`
`() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 2621).

6.298.3.3 `virtual void ac-`
`tivemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller::looseMarshal`
`(OpenWireFormat * wireFormat, commands::DataStructure *`
`dataStructure, decaf::io::DataOutputStream * dataOut) throw (`
`decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 2621).

6.298.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 2622).

6.298.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 2622).

```

6.298.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 2623).

```

6.298.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller::tightUnmarsh
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]

```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 2623).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ExceptionResponseMarshaller.h`

6.299 `activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller` Class Reference

Marshaling code for Open Wire Format for `ExceptionResponseMarshaller` (p. 1491).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ExceptionResponseMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller`:

Public Member Functions

- `ExceptionResponseMarshaller ()`
- `virtual ~ExceptionResponseMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaller.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write a object instance to data output stream.
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)`
Write a object instance to data output stream.

6.299.1 Detailed Description

Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p.1491). NOTE! This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.299.2 Constructor & Destructor Documentation

6.299.2.1 `activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller::ExceptionRe`
`() [inline]`

6.299.2.2 `virtual`
`activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller::~~ExceptionR`
`() [inline, virtual]`

6.299.3 Member Function Documentation

6.299.3.1 `virtual commands::DataStructure* ac-`
`tivemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller::createObject`
`() const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 2634).

6.299.3.2 `virtual unsigned char ac-`
`tivemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller::getDataStructu`
`() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 2635).

6.299.3.3 `virtual void ac-`
`tivemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller::looseMarshal`
`(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (`
`decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 2635).

6.299.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 2635).

6.299.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 2636).

```

6.299.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 2636).

```

6.299.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller::tightUnmarsh
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]

```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 2637).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/ExceptionResponseMarshaller.h`

6.300 activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1495).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ExceptionResponseMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller:

Public Member Functions

- **ExceptionResponseMarshaller** ()
• virtual **~ExceptionResponseMarshaller** ()
• virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.300.1 Detailed Description

Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p.1495). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.300.2 Constructor & Destructor Documentation

6.300.2.1 `activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller::ExceptionRe`
`() [inline]`

6.300.2.2 `virtual`
`activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller::~~ExceptionR`
`() [inline, virtual]`

6.300.3 Member Function Documentation

6.300.3.1 `virtual commands::DataStructure* ac-`
`tivemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller::createObject`
`() const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 2625).

6.300.3.2 `virtual unsigned char ac-`
`tivemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller::getDataStructu`
`() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 2626).

6.300.3.3 `virtual void ac-`
`tivemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller::looseMarshal`
`(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (`
`decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 2626).

6.300.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 2626).

6.300.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 2627).

```

6.300.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 2627).

```

6.300.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller::tightUnmarsh
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]

```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 2628).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ExceptionResponseMarshaller.h`

6.301 activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1499).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ExceptionResponseMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller:

Public Member Functions

- **ExceptionResponseMarshaller** ()
• virtual **~ExceptionResponseMarshaller** ()
• virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.301.1 Detailed Description

Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p.1499). NOTE! This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.301.2 Constructor & Destructor Documentation

6.301.2.1 `activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller::ExceptionRe`
`() [inline]`

6.301.2.2 `virtual`
`activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller::~~ExceptionR`
`() [inline, virtual]`

6.301.3 Member Function Documentation

6.301.3.1 `virtual commands::DataStructure* ac-`
`tivemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller::createObject`
`() const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 2639).

6.301.3.2 `virtual unsigned char ac-`
`tivemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller::getDataStructu`
`() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 2639).

6.301.3.3 `virtual void ac-`
`tivemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller::looseMarshal`
`(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (`
`decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 2639).

6.301.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 2640).

6.301.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 2640).

```

6.301.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 2641).

```

6.301.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller::tightUnmarsh
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]

```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 2641).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/ExceptionResponseMarshaller.h`

6.302 activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1503).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ExceptionResponseMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller:

Public Member Functions

- **ExceptionResponseMarshaller** ()
- virtual **~ExceptionResponseMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.302.1 Detailed Description

Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p.1503). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.302.2 Constructor & Destructor Documentation

6.302.2.1 `activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller::ExceptionRe`
`() [inline]`

6.302.2.2 `virtual`
`activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller::~~ExceptionR`
`() [inline, virtual]`

6.302.3 Member Function Documentation

6.302.3.1 `virtual commands::DataStructure* ac-`
`tivemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller::createObject`
`() const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 2630).

6.302.3.2 `virtual unsigned char ac-`
`tivemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller::getDataStructu`
`() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 2630).

6.302.3.3 `virtual void ac-`
`tivemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller::looseMarshal`
`(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (`
`decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 2630).

6.302.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 2631).

6.302.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 2631).

```

6.302.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 2632).

```

6.302.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller::tightUnmarsh
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]

```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 2632).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ExceptionResponseMarshaller.h`

6.303 decaf::util::concurrent::ExecutionException Class Reference

```
#include <src/main/decaf/util/concurrent/ExecutionException.h>
```

Inheritance diagram for decaf::util::concurrent::ExecutionException:

Public Member Functions

- **ExecutionException** () throw ()
Default Constructor.
- **ExecutionException** (const **decaf::lang::Exception** &ex) throw ()
Conversion Constructor from some other Exception.
- **ExecutionException** (const **ExecutionException** &ex) throw ()
Copy Constructor.
- **ExecutionException** (const std::exception *cause) throw ()
Constructor.
- **ExecutionException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **ExecutionException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **ExecutionException** * clone () const
Clones this exception.
- virtual ~**ExecutionException** () throw ()

6.303.1 Constructor & Destructor Documentation

6.303.1.1 decaf::util::concurrent::ExecutionException::ExecutionException () throw () [inline]

Default Constructor.

6.303.1.2 decaf::util::concurrent::ExecutionException::ExecutionException (const decaf::lang::Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

Parameters

ex - An exception that should become this type of Exception

References `decaf::lang::Exception::Exception()`.

6.303.1.3 `decaf::util::concurrent::ExecutionException::ExecutionException (const ExecutionException & ex) throw () [inline]`

Copy Constructor.

Parameters

ex - The Exception to copy in this new instance.

References `decaf::lang::Exception::Exception()`.

6.303.1.4 `decaf::util::concurrent::ExecutionException::ExecutionException (const std::exception * cause) throw () [inline]`

Constructor.

Parameters

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.303.1.5 `decaf::util::concurrent::ExecutionException::ExecutionException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file - The file name where exception occurs

lineNumber - The line number where the exception occurred.

msg - The message to report

... - The list of primitives that are formatted into the message

6.303.1.6 `decaf::util::concurrent::ExecutionException::ExecutionException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file - The file name where exception occurs

lineNumber - The line number where the exception occurred.

cause - The exception that was the cause for this one to be thrown.

msg - The message to report

... - list of primitives that are formatted into the message

6.303.1.7 virtual decaf::util::concurrent::ExecutionException::~~ExecutionException () throw () [inline, virtual]

6.303.2 Member Function Documentation

6.303.2.1 virtual ExecutionException* decaf::util::concurrent::ExecutionException::clone () const [inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

a new instance of an exception that is a clone of this one.

Reimplemented from **decaf::lang::Exception** (p. 1479).

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**ExecutionException.h**

6.304 decaf::util::concurrent::Executor Class Reference

An object that executes submitted **decaf.lang Runnable** (p. 2642) tasks.

```
#include <src/main/decaf/util/concurrent/Executor.h>
```

Inheritance diagram for decaf::util::concurrent::Executor:

Public Member Functions

- virtual ~**Executor** ()
- virtual void **execute** (Runnable *command)=0 throw (decaf::util::concurrent::RejectedExecutionException, decaf::lang::exceptions::NullPointerException)

Executes the given command at some time in the future.

6.304.1 Detailed Description

An object that executes submitted **decaf.lang Runnable** (p. 2642) tasks. This interface provides a way of decoupling task submission from the mechanics of how each task will be run, including details of thread use, scheduling, etc. An **Executor** (p. 1509) is normally used instead of explicitly creating threads. For example, rather than invoking `new Thread(new RunnableTask()).start()` for each of a set of tasks, you might use:

```
Executor (p. 1509) executor = anExecutor;
```

```

executor->execute( new RunnableTask1() );
executor->execute( new RunnableTask2() );
...

```

However, the `Executor` (p.1509) interface does not strictly require that execution be asynchronous. In the simplest case, an executor can run the submitted task immediately in the caller's thread:

```

class DirectExecutor : public Executor (p.1509) {
public:

    void execute( Runnable* r ) (p.1510) {
        r->run();
    }

}

```

More typically, tasks are executed in some thread other than the caller's thread. The executor below spawns a new thread for each task.

```

class ThreadPerTaskExecutor : public Executor (p.1509) {
public:
    std::vector<Thread*gt; threads;

    void execute( Runnable* r ) (p.1510) {
        threads.push_back( new Thread( r ) );
        threads.rbegin()->start();
    }

}

```

The `Executor` (p.1509) implementations provided in this package implement **decaf.util.concurrent.ExecutorService** (p.1511), which is a more extensive interface. The **decaf.util.concurrent.ThreadPoolExecutor** (p.??) class provides an extensible thread pool implementation. The **decaf.util.concurrent.Executor** (p.??) class provides convenient factory methods for these Executors.

Since

1.0

6.304.2 Constructor & Destructor Documentation

6.304.2.1 `virtual decaf::util::concurrent::Executor::~~Executor () [inline, virtual]`

6.304.3 Member Function Documentation

6.304.3.1 `virtual void decaf::util::concurrent::Executor::execute (Runnable * command) throw (decaf::util::concurrent::RejectedExecutionException, decaf::lang::exceptions::NullPointerException) [pure virtual]`

Executes the given command at some time in the future.

The command may execute in a new thread, in a pooled thread, or in the calling thread, at the discretion of the **Executor** (p. 1509) implementation.

Parameters

command the runnable task

Exceptions

RejectedExecutionException (p. 2533) if this task cannot be accepted for execution.

NullPointerException if command is null

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**Executor.h**

6.305 decaf::util::concurrent::ExecutorService Class Reference

An **Executor** (p. 1509) that provides methods to manage termination and methods that can produce a **Future** (p. 1597) for tracking progress of one or more asynchronous tasks.

```
#include <src/main/decaf/util/concurrent/ExecutorService.h>
```

Inheritance diagram for decaf::util::concurrent::ExecutorService:

Public Member Functions

- virtual **~ExecutorService** ()
- bool **awaitTermination** (long long timeout, const **TimeUnit** &unit)=0 throw (decaf::lang::exceptions::InterruptedException)
Blocks until all tasks have completed execution after a shutdown request, or the timeout occurs, or the current thread is interrupted, whichever happens first.

6.305.1 Detailed Description

An **Executor** (p. 1509) that provides methods to manage termination and methods that can produce a **Future** (p. 1597) for tracking progress of one or more asynchronous tasks. An **ExecutorService** (p. 1511) can be shut down, which will cause it to reject new tasks. Two different methods are provided for shutting down an **ExecutorService** (p. 1511). The shutdown() method will allow previously submitted tasks to execute before terminating, while the shutdownNow() method prevents waiting tasks from starting and attempts to stop currently executing tasks. Upon termination, an executor has no tasks actively executing, no tasks awaiting execution, and no new tasks can be submitted. An unused **ExecutorService** (p. 1511) should be shut down to allow reclamation of its resources.

Method submit extends base method **Executor.execute** (p. 1510)(decaf.lang.**Runnable** (p. 2642)) by creating and returning a **Future** (p. 1597) that can be used to cancel execution

and/or wait for completion. Methods `invokeAny` and `invokeAll` perform the most commonly useful forms of bulk execution, executing a collection of tasks and then waiting for at least one, or all, to complete. (Class `ExecutorCompletionService` can be used to write customized variants of these methods.)

The `Executors` class provides factory methods for the executor services provided in this package.

Since

1.0

6.305.2 Constructor & Destructor Documentation

6.305.2.1 `virtual decaf::util::concurrent::ExecutorService::~~ExecutorService ()`
[inline, virtual]

6.305.3 Member Function Documentation

6.305.3.1 `bool decaf::util::concurrent::ExecutorService::awaitTermination`
(`long long timeout`, `const TimeUnit & unit`) `throw (`
`decaf::lang::exceptions::InterruptedException)` [pure virtual]

Blocks until all tasks have completed execution after a shutdown request, or the timeout occurs, or the current thread is interrupted, whichever happens first.

Parameters

timeout The amount of time to wait before timing out the Wait operation.

unit The Units that comprise the timeout value.

Returns

true if the executor terminated before the given timeout value elapsed.

Exceptions

InterruptedException - if interrupted while waiting.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/ExecutorService.h`

6.306 activemq::transport::failover::FailoverTransport Class Reference

```
#include <src/main/activemq/transport/failover/FailoverTransport.h>
```

Inheritance diagram for `activemq::transport::failover::FailoverTransport`:

Public Member Functions

- **FailoverTransport** ()
- virtual **~FailoverTransport** ()
- void **reconnect** ()
*Indicates that the **Transport** (p. 3066) needs to reconnect to another URI in its list.*
- void **add** (const std::string &uri)
Adds a New URI to the List of URIs this transport can Connect to.
- virtual void **addURI** (const List< URI > &uris)
*Add a URI to the list of URI's that will represent the set of Transports that this **Transport** (p. 3066) is a composite of.*
- virtual void **removeURI** (const List< URI > &uris)
*Remove a URI from the set of URI's that represents the set of Transports that this **Transport** (p. 3066) is composed of, removing a URI for which the composite has created a connected **Transport** (p. 3066) should result in that **Transport** (p. 3066) being disposed of.*
- virtual void **start** () throw (decaf::io::IOException)
Starts this transport object and creates the thread for polling on the input stream for commands.
- virtual void **stop** () throw (decaf::io::IOException)
*Stop the **Transport** (p. 3066).*
- virtual void **close** () throw (decaf::io::IOException)
Stops the polling thread and closes the streams.
- virtual void **oneway** (const Pointer< Command > &command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)
Sends a one-way command.
- virtual Pointer< Response > **request** (const Pointer< Command > &command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)
Sends the given command to the broker and then waits for the response.
- virtual Pointer< Response > **request** (const Pointer< Command > &command, unsigned int timeout) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)
Sends the given command to the broker and then waits for the response.
- virtual void **setWireFormat** (const Pointer< wireformat::WireFormat > &wireFormat AMQCPP_UNUSED)
Sets the WireFormat instance to use.
- virtual void **setTransportListener** (TransportListener *listener)
Sets the observer of asynchronous events from this transport.
- virtual TransportListener * **getTransportListener** () const
Gets the observer of asynchronous exceptions from this transport.

- virtual bool **isFaultTolerant** () const
*Is this **Transport** (p. 3066) fault tolerant, meaning that it will reconnect to a broker on disconnect.*
- virtual bool **isConnected** () const
*Is the **Transport** (p. 3066) Connected to its Broker.*
- virtual bool **isClosed** () const
*Has the **Transport** (p. 3066) been shutdown and no longer usable.*
- bool **isInitialized** () const
*Returns true if the **Transport** (p. 3066) has been initialized by a BrokerInfo command.*
- void **setInitialized** (bool value)
*Sets the initialized state of this **Transport** (p. 3066) to true.*
- virtual **Transport * narrow** (const std::type_info &typeId)
*Narrows down a Chain of Transports to a specific **Transport** (p. 3066) to allow a higher level transport to skip intermediate Transports in certain circumstances.*
- virtual std::string **getRemoteAddress** () const
- virtual bool **isPending** () const
- virtual bool **iterate** ()
*Performs the actual Reconnect operation for the **FailoverTransport** (p. 1512), when a connection is made this method returns false to indicate it doesn't need to run again, otherwise it returns true to indicate its still trying to connect.*
- virtual void **reconnect** (const **decaf::net::URI** &uri) throw (**decaf::io::IOException**)
reconnect to another location
- long long **getTimeout** () const
- void **setTimeout** (long long value)
- long long **getInitialReconnectDelay** () const
- void **setInitialReconnectDelay** (long long value)
- long long **getMaxReconnectDelay** () const
- void **setMaxReconnectDelay** (long long value)
- long long **getBackOffMultiplier** () const
- void **setBackOffMultiplier** (long long value)
- bool **isUseExponentialBackOff** () const
- void **setUseExponentialBackOff** (bool value)
- bool **isRandomize** () const
- void **setRandomize** (bool value)
- int **getMaxReconnectAttempts** () const
- void **setMaxReconnectAttempts** (int value)
- long long **getReconnectDelay** () const
- void **setReconnectDelay** (long long value)
- bool **isBackup** () const
- void **setBackup** (bool value)
- int **getBackupPoolSize** () const
- void **setBackupPoolSize** (int value)
- bool **isTrackMessages** () const

- void **setTrackMessages** (bool value)
- int **getMaxCacheSize** () const
- void **setMaxCacheSize** (int value)

Protected Member Functions

- void **restoreTransport** (const **Pointer**< **Transport** > &transport) throw (decaf::io::IOException)
*Given a **Transport** (p. 3066) restore the state of the Client's connection to the Broker using the data accumulated in the State Tracker.*
- void **handleTransportFailure** (const decaf::lang::Exception &error) throw (decaf::lang::Exception)
*Called when this class' **TransportListener** (p. 3080) is notified of a Failure.*

Friends

- class **FailoverTransportListener**

6.306.1 Constructor & Destructor Documentation

6.306.1.1 **activemq::transport::failover::FailoverTransport::FailoverTransport** ()

6.306.1.2 **virtual**
activemq::transport::failover::FailoverTransport::~~FailoverTransport ()
 [virtual]

6.306.2 Member Function Documentation

6.306.2.1 **void** **activemq::transport::failover::FailoverTransport::add** (const
 std::string & *uri*)

Adds a New URI to the List of URIs this transport can Connect to.

Parameters

uri A String version of a URI to add to the URIs to failover to.

6.306.2.2 **virtual void** **activemq::transport::failover::FailoverTransport::addURI** (const List< URI > & *uris*) [virtual]

Add a URI to the list of URI's that will represent the set of Transports that this **Transport** (p. 3066) is a composite of.

Parameters

uris The new URIs to add to the set this composite maintains.

6.306.2.3 `virtual void activemq::transport::failover::FailoverTransport::close ()
throw (decaf::io::IOException) [virtual]`

Stops the polling thread and closes the streams.

This can be called explicitly, but is also called in the destructor. Once this object has been closed, it cannot be restarted.

Exceptions

IOException if errors occur.

6.306.2.4 `long long activemq::transport::failover::FailoverTransport::getBackOffMultiplier ()
const [inline]`

6.306.2.5 `int activemq::transport::failover::FailoverTransport::getBackupPoolSize ()
const [inline]`

6.306.2.6 `long long activemq::transport::failover::FailoverTransport::getInitialReconnectDelay ()
const [inline]`

6.306.2.7 `int activemq::transport::failover::FailoverTransport::getMaxCacheSize ()
const [inline]`

6.306.2.8 `int activemq::transport::failover::FailoverTransport::getMaxReconnectAttempts ()
const [inline]`

6.306.2.9 `long long activemq::transport::failover::FailoverTransport::getMaxReconnectDelay ()
const [inline]`

6.306.2.10 `long long activemq::transport::failover::FailoverTransport::getReconnectDelay ()
const [inline]`

6.306.2.11 `virtual std::string activemq::transport::failover::FailoverTransport::getRemoteAddress ()
const [virtual]`

Returns

the remote address for this connection

6.306.2.12 `long long activemq::transport::failover::FailoverTransport::getTimeout ()
const [inline]`

6.306.2.13 `virtual TransportListener* activemq::transport::failover::FailoverTransport::getTransportListener ()
const [virtual]`

Gets the observer of asynchronous exceptions from this transport.

Returns

The listener of transport events.

6.306.2.14 `void activemq::transport::failover::FailoverTransport::handleTransportFailure (const decaf::lang::Exception & error) throw (decaf::lang::Exception) [protected]`

Called when this class' **TransportListener** (p. 3080) is notified of a Failure.

Parameters

error - The CMS Exception that was thrown.

Exceptions

Exception if an error occurs.

6.306.2.15 `bool activemq::transport::failover::FailoverTransport::isBackup () const [inline]`

6.306.2.16 `virtual bool activemq::transport::failover::FailoverTransport::isClosed () const [inline, virtual]`

Has the **Transport** (p. 3066) been shutdown and no longer usable.

Returns

true if the **Transport** (p. 3066)

6.306.2.17 `virtual bool activemq::transport::failover::FailoverTransport::isConnected () const [inline, virtual]`

Is the **Transport** (p. 3066) Connected to its Broker.

Returns

true if a connection has been made.

6.306.2.18 `virtual bool activemq::transport::failover::FailoverTransport::isFaultTolerant () const [inline, virtual]`

Is this **Transport** (p. 3066) fault tolerant, meaning that it will reconnect to a broker on disconnect.

Returns

true if the **Transport** (p. 3066) is fault tolerant.

6.306.2.19 `bool activemq::transport::failover::FailoverTransport::isInitialized ()`
`const [inline]`

Returns true if the **Transport** (p. 3066) has been initialized by a BrokerInfo command.

Returns

true if the **Transport** (p. 3066) has been initialized by a BrokerInfo command.

6.306.2.20 `virtual bool activemq::transport::failover::FailoverTransport::isPending () const [virtual]`

Returns

true if there is a need for the iterate method to be called by this classes task runner.

Implements **activemq::threads::CompositeTask** (p. 1016).

6.306.2.21 `bool activemq::transport::failover::FailoverTransport::isRandomize ()`
`const [inline]`

6.306.2.22 `bool activemq::transport::failover::FailoverTransport::isTrackMessages () const [inline]`

6.306.2.23 `bool activemq::transport::failover::FailoverTransport::isUseExponentialBackOff () const [inline]`

6.306.2.24 `virtual bool activemq::transport::failover::FailoverTransport::iterate () [virtual]`

Performs the actual Reconnect operation for the **FailoverTransport** (p. 1512), when a connection is made this method returns false to indicate it doesn't need to run again, otherwise it returns true to indicate its still trying to connect.

Returns

false to indicate a connection, true to indicate it needs to try again.

Implements **activemq::threads::Task** (p. 2956).

6.306.2.25 `virtual Transport* activemq::transport::failover::FailoverTransport::narrow (const std::type_info & typeId) [inline, virtual]`

Narrows down a Chain of Transports to a specific **Transport** (p. 3066) to allow a higher level transport to skip intermediate Transports in certain circumstances.

Parameters

typeId - The type_info of the Object we are searching for.

Returns

the requested Object. or NULL if its not in this chain.

References `activemq::transport::Transport::narrow()`.

6.306.2.26 `virtual void activemq::transport::failover::FailoverTransport::oneway
(const Pointer< Command > & command
) throw (decaf::io::IOException, de-
caf::lang::exceptions::UnsupportedOperationException)
[virtual]`

Sends a one-way command.

Does not wait for any response from the broker.

Parameters

command the command to be sent.

Exceptions

IOException if an exception occurs during writing of the command.

UnsupportedOperationException if this method is not implemented by this transport.

6.306.2.27 `void activemq::transport::failover::FailoverTransport::reconnect ()`

Indicates that the **Transport** (p. 3066) needs to reconnect to another URI in its list.

6.306.2.28 `virtual void activemq::transport::failover::FailoverTransport::reconnect
(const decaf::net::URI & uri) throw (decaf::io::IOException)
[virtual]`

reconnect to another location

Parameters

uri

Exceptions

IOException on failure of if not supported

6.306.2.29 `virtual void activemq::transport::failover::FailoverTransport::removeURI
(const List< URI > & uris) [virtual]`

Remove a URI from the set of URI's that represents the set of Transports that this **Transport** (p. 3066) is composed of, removing a URI for which the composite has created a connected **Transport** (p. 3066) should result in that **Transport** (p. 3066) being disposed of.

Parameters

uris The new URIs to remove to the set this composite maintains.

6.306.2.30 `virtual Pointer<Response> activemq::transport::failover::FailoverTransport::request (const Pointer< Command > & command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Sends the given command to the broker and then waits for the response.

Parameters

command the command to be sent.

Returns

the response from the broker.

Exceptions

IOException if an exception occurs during the read of the command.

UnsupportedOperationException if this method is not implemented by this transport.

6.306.2.31 `virtual Pointer<Response> activemq::transport::failover::FailoverTransport::request (const Pointer< Command > & command, unsigned int timeout) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Sends the given command to the broker and then waits for the response.

Parameters

command - The command to be sent.

timeout - The time to wait for this response.

Returns

the response from the broker.

Exceptions

IOException if an exception occurs during the read of the command.

UnsupportedOperationException if this method is not implemented by this transport.

6.306.2.32 `void activemq::transport::failover::FailoverTransport::restoreTransport (const Pointer< Transport > & transport) throw (decaf::io::IOException) [protected]`

Given a **Transport** (p. 3066) restore the state of the Client's connection to the Broker using the data accumulated in the State Tracker.

Parameters

transport The new **Transport** (p. 3066) connected to the Broker.

Exceptions

IOException if an errors occurs while restoring the old state.

- 6.306.2.33** void activemq::transport::failover::FailoverTransport::setBackOffMultiplier (long long *value*) [inline]
- 6.306.2.34** void activemq::transport::failover::FailoverTransport::setBackup (bool *value*) [inline]
- 6.306.2.35** void activemq::transport::failover::FailoverTransport::setBackupPoolSize (int *value*) [inline]
- 6.306.2.36** void activemq::transport::failover::FailoverTransport::setInitialized (bool *value*) [inline]

Sets the initialized state of this **Transport** (p. 3066) to true.

Parameters

value - true if this **Transport** (p. 3066) has been initialized.

- 6.306.2.37** void activemq::transport::failover::FailoverTransport::setInitialReconnectDelay (long long *value*) [inline]
- 6.306.2.38** void activemq::transport::failover::FailoverTransport::setMaxCacheSize (int *value*) [inline]
- 6.306.2.39** void activemq::transport::failover::FailoverTransport::setMaxReconnectAttempts (int *value*) [inline]
- 6.306.2.40** void activemq::transport::failover::FailoverTransport::setMaxReconnectDelay (long long *value*) [inline]
- 6.306.2.41** void activemq::transport::failover::FailoverTransport::setRandomize (bool *value*) [inline]
- 6.306.2.42** void activemq::transport::failover::FailoverTransport::setReconnectDelay (long long *value*) [inline]
- 6.306.2.43** void activemq::transport::failover::FailoverTransport::setTimeout (long long *value*) [inline]
- 6.306.2.44** void activemq::transport::failover::FailoverTransport::setTrackMessages (bool *value*) [inline]
- 6.306.2.45** virtual void activemq::transport::failover::FailoverTransport::setTransportListener (TransportListener * *listener*) [virtual]

Sets the observer of asynchronous events from this transport.

Parameters

listener the listener of transport events.

6.306.2.46 `void activemq::transport::failover::FailoverTransport::setUseExponentialBackOff (bool value) [inline]`

6.306.2.47 `virtual void activemq::transport::failover::FailoverTransport::setWireFormat (const Pointer< wireformat::WireFormat > &wireFormat AMQCPP_UNUSED) [inline, virtual]`

Sets the WireFormat instance to use.

Parameters

wireFormat The WireFormat the object used to encode / decode commands.

6.306.2.48 `virtual void activemq::transport::failover::FailoverTransport::start () throw (decaf::io::IOException) [virtual]`

Starts this transport object and creates the thread for polling on the input stream for commands.

If this object has been closed, throws an exception. Before calling start, the caller must set the IO streams and the reader and writer objects.

Exceptions

IOException if an error occurs or if this transport has already been closed.

6.306.2.49 `virtual void activemq::transport::failover::FailoverTransport::stop () throw (decaf::io::IOException) [virtual]`

Stop the **Transport** (p. 3066).

Exceptions

IOException if an error occurs while stopping the **Transport** (p. 3066).

6.306.3 Friends And Related Function Documentation

6.306.3.1 `friend class FailoverTransportListener [friend]`

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/failover/FailoverTransport.h`

6.307 activemq::transport::failover::FailoverTransportFactory Class Reference

Creates an instance of a **FailoverTransport** (p.1512).

```
#include <src/main/activemq/transport/failover/FailoverTransportFactory.h>
```

Inheritance diagram for activemq::transport::failover::FailoverTransportFactory:

Public Member Functions

- virtual **~FailoverTransportFactory** ()
- virtual **Pointer< Transport > create** (const **decaf::net::URI** &location) throw (exceptions::ActiveMQException)

*Creates a fully configured **Transport** (p.3066) instance which could be a chain of filters and transports.*

- virtual **Pointer< Transport > createComposite** (const **decaf::net::URI** &location) throw (exceptions::ActiveMQException)

*Creates a slimed down **Transport** (p.3066) instance which can be used in composite transport instances.*

Protected Member Functions

- virtual **Pointer< Transport > doCreateComposite** (const **decaf::net::URI** &location, const **decaf::util::Properties** &properties) throw (exceptions::ActiveMQException)

*Creates a slimed down **Transport** (p.3066) instance which can be used in composite transport instances.*

6.307.1 Detailed Description

Creates an instance of a **FailoverTransport** (p.1512).

Since

3.0

6.307.2 Constructor & Destructor Documentation

6.307.2.1 `virtual
activemq::transport::failover::FailoverTransportFactory::~FailoverTransportFactory
() [inline, virtual]`

6.307.3 Member Function Documentation

6.307.3.1 `virtual Pointer<Transport> ac-
tivemq::transport::failover::FailoverTransportFactory::create (const
decaf::net::URI & location) throw (exceptions::ActiveMQException)
[virtual]`

Creates a fully configured **Transport** (p.3066) instance which could be a chain of filters and transports.

Parameters

location - URI location to connect to plus any properties to assign.

Exceptions

ActiveMQException if an error occurs

Implements `activemq::transport::TransportFactory` (p.3072).

6.307.3.2 `virtual Pointer<Transport> ac-
tivemq::transport::failover::FailoverTransportFactory::createComposite
(const decaf::net::URI & location) throw (exceptions::ActiveMQException) [virtual]`

Creates a slimed down **Transport** (p.3066) instance which can be used in composite transport instances.

Parameters

location - URI location to connect to plus any properties to assign.

Exceptions

ActiveMQException if an error occurs

Implements `activemq::transport::TransportFactory` (p.3072).

6.307.3.3 `virtual Pointer<Transport> ac-
tivemq::transport::failover::FailoverTransportFactory::doCreateComposite
(const decaf::net::URI & location, const decaf::util::Properties &
properties) throw (exceptions::ActiveMQException) [protected,
virtual]`

Creates a slimed down **Transport** (p.3066) instance which can be used in composite transport instances.

Parameters

- location* - URI location to connect to.
- properties* - Properties to apply to the transport.

Returns

Pointer to a new **FailoverTransport** (p. 1512) instance.

Exceptions

- ActiveMQException* if an error occurs

The documentation for this class was generated from the following file:

- src/main/activemq/transport/failover/**FailoverTransportFactory.h**

6.308 activemq::transport::failover::FailoverTransportListener Class Reference

Utility class used by the **Transport** (p. 3066) to perform the work of responding to events from the active **Transport** (p. 3066).

```
#include <src/main/activemq/transport/failover/FailoverTransportListener.h>
```

Inheritance diagram for activemq::transport::failover::FailoverTransportListener:

Public Member Functions

- **FailoverTransportListener** (**FailoverTransport** *parent)
- virtual **~FailoverTransportListener** ()
- virtual void **onCommand** (const **Pointer**< **Command** > &command)
Event handler for the receipt of a command.
- virtual void **onException** (const **decaf::lang::Exception** &ex)
Event handler for an exception from a command transport.
- virtual void **transportInterrupted** ()
The transport has suffered an interruption from which it hopes to recover.
- virtual void **transportResumed** ()
The transport has resumed after an interruption.

6.308.1 Detailed Description

Utility class used by the **Transport** (p. 3066) to perform the work of responding to events from the active **Transport** (p. 3066).

Since

3.0

6.308.2 Constructor & Destructor Documentation

6.308.2.1 `activemq::transport::failover::FailoverTransportListener::FailoverTransportListener (FailoverTransport * parent)`

6.308.2.2 `virtual
activemq::transport::failover::FailoverTransportListener::~~FailoverTransportListener
() [virtual]`

6.308.3 Member Function Documentation

6.308.3.1 `virtual void ac-
tivemq::transport::failover::FailoverTransportListener::onCommand (
const Pointer< Command > & command) [virtual]`

Event handler for the receipt of a command.

The transport passes off all received commands to its listeners, the listener then owns the Object. If there is no registered listener the **Transport** (p. 3066) deletes the command upon receipt.

Parameters

command the received command object.

Implements `activemq::transport::TransportListener` (p. 3081).

6.308.3.2 `virtual void ac-
tivemq::transport::failover::FailoverTransportListener::onException (
const decaf::lang::Exception & ex) [virtual]`

Event handler for an exception from a command transport.

Parameters

ex The exception.

Implements `activemq::transport::TransportListener` (p. 3082).

6.308.3.3 `virtual void ac-
tivemq::transport::failover::FailoverTransportListener::transportInterrupted
() [virtual]`

The transport has suffered an interruption from which it hopes to recover.

Implements `activemq::transport::TransportListener` (p. 3082).

6.308.3.4 `virtual void ac-
tivemq::transport::failover::FailoverTransportListener::transportResumed
() [virtual]`

The transport has resumed after an interruption.

Implements `activemq::transport::TransportListener` (p. 3082).

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/failover/FailoverTransportListener.h`

6.309 decaf::util::logging::Filter Class Reference

A **Filter** (p. 1527) can be used to provide fine grain control over what is logged, beyond the control provided by log levels.

```
#include <src/main/decaf/util/logging/Filter.h>
```

Public Member Functions

- virtual `~Filter ()`
- virtual `bool isLoggable (const LogRecord &record) const =0`

Check if a given log record should be published.

6.309.1 Detailed Description

A **Filter** (p.1527) can be used to provide fine grain control over what is logged, beyond the control provided by log levels. Each **Logger** (p.1908) and each **Handler** (p.1604) can have a filter associated with it. The **Logger** (p.1908) or **Handler** (p.1604) will call the `isLoggable` method to check if a given **LogRecord** (p.1928) should be published. If `isLoggable` returns false, the **LogRecord** (p.1928) will be discarded.

6.309.2 Constructor & Destructor Documentation

6.309.2.1 virtual `decaf::util::logging::Filter::~Filter ()` [inline, virtual]

6.309.3 Member Function Documentation

6.309.3.1 virtual `bool decaf::util::logging::Filter::isLoggable (const LogRecord &
record) const` [pure virtual]

Check if a given log record should be published.

Parameters

record the **LogRecord** (p.1928) to check.

Returns

true if the record is loggable.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/logging/Filter.h`

6.310 decaf::io::FilterInputStream Class Reference

A **FilterInputStream** (p.1528) contains some other input stream, which it uses as its basic source of data, possibly transforming the data along the way or providing additional functionality.

```
#include <src/main/decaf/io/FilterInputStream.h>
```

Inheritance diagram for decaf::io::FilterInputStream:

Public Member Functions

- **FilterInputStream** (**InputStream** *inputStream, bool own=false)
*Constructor to create a wrapped **InputStream** (p. 1630).*
- virtual ~**FilterInputStream** ()
- virtual std::size_t **available** () const throw (**IOException**)
Returns the number of bytes that can be read from this input stream without blocking.
- virtual int **read** () throw (**IOException**)
Reads the next byte of data from this input stream.
- virtual int **read** (unsigned char *buffer, std::size_t offset, std::size_t bufferSize) throw (**IOException**, lang::exceptions::NullPointerException)
Reads up to len bytes of data from this input stream into an array of bytes.
- virtual void **close** () throw (io::IOException)
*Close the Stream, the **FilterOutputStream** (p. 1536) simply calls the close method of the underlying stream.*
- virtual std::size_t **skip** (std::size_t num) throw (io::IOException, lang::exceptions::UnsupportedOperationException)
Skips over and discards n bytes of data from this input stream.
- virtual void **mark** (int readLimit)
Marks the current position in the stream A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.
- virtual void **reset** () throw (**IOException**)
Repositions this stream to the position at the time the mark method was last called on this input stream.
- virtual bool **markSupported** () const
Determines if this input stream supports the mark and reset methods.
- virtual void **lock** () throw (decaf::lang::exceptions::RuntimeException)
Locks the object.
- virtual bool **tryLock** () throw (decaf::lang::exceptions::RuntimeException)
Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

- virtual void **unlock** () throw (decaf::lang::exceptions::RuntimeException)
Unlocks the object.
- virtual void **wait** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (long long millisecs) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (long long millisecs, int nanos) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **notify** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)
Signals a waiter on this object that it can now wake up and continue.
- virtual void **notifyAll** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)
Signals the waiters on this object that it can now wake up and continue.

Protected Member Functions

- virtual bool **isClosed** () const

Protected Attributes

- **InputStream * inputStream**
- **util::concurrent::Mutex mutex**
- bool **own**
- volatile bool **closed**

6.310.1 Detailed Description

A **FilterInputStream** (p. 1528) contains some other input stream, which it uses as its basic source of data, possibly transforming the data along the way or providing additional functionality. The class **FilterInputStream** (p. 1528) itself simply overrides all methods of **InputStream** (p. 1630) with versions that pass all requests to the contained input stream. Subclasses of **FilterInputStream** (p. 1528) may further override some of these methods and may also provide additional methods and fields.

6.310.2 Constructor & Destructor Documentation

6.310.2.1 `decaf::io::FilterInputStream::FilterInputStream (InputStream * inputStream, bool own = false)` [inline]

Constructor to create a wrapped **InputStream** (p.1630).

Parameters

inputStream the stream to wrap and filter

own indicates if we own the stream object, defaults to false

6.310.2.2 `virtual decaf::io::FilterInputStream::~FilterInputStream ()` [inline, virtual]

References DECAF_CATCH_NOTHROW, and DECAF_CATCHALL_NOTHROW.

6.310.3 Member Function Documentation

6.310.3.1 `virtual std::size_t decaf::io::FilterInputStream::available () const`
`throw (IOException)` [inline, virtual]

Returns the number of bytes that can be read from this input stream without blocking.

This method simply performs in.available() and returns the result.

Returns

the number of bytes available without blocking.

Implements **decaf::io::InputStream** (p.1631).

Reimplemented in **decaf::io::BufferedInputStream** (p.749).

References DECAF_CATCH_RETHROW, and DECAF_CATCHALL_THROW.

6.310.3.2 `virtual void decaf::io::FilterInputStream::close ()` `throw (io::IOException)` [inline, virtual]

Close the Stream, the **FilterOutputStream** (p.1536) simply calls the close method of the underlying stream.

Exceptions

Exception

Reimplemented in **decaf::io::BufferedInputStream** (p.749).

References DECAF_CATCH_RETHROW, and DECAF_CATCHALL_THROW.

6.310.3.3 `virtual bool decaf::io::FilterInputStream::isClosed () const` [inline, protected, virtual]

Returns

true if this stream has been closed.

6.310.3.4 `virtual void decaf::io::FilterInputStream::lock () throw (decaf::lang::exceptions::RuntimeException)` [inline, virtual]

Locks the object.

Exceptions

RuntimeException if an error occurs while locking the object.

Implements `decaf::util::concurrent::Synchronizable` (p. 2932).

6.310.3.5 `virtual void decaf::io::FilterInputStream::mark (int readLimit)` [inline, virtual]

Marks the current position in the stream. A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.

If a stream instance reports that marks are supported then the stream will ensure that the same bytes can be read again after the reset method is called so long the readLimit is not reached.

Parameters

readLimit The max bytes read before marked position is invalid.

Implements `decaf::io::InputStream` (p. 1631).

Reimplemented in `decaf::io::BufferedInputStream` (p. 750).

References `DECAF_CATCHALL_NOTHROW`.

6.310.3.6 `virtual bool decaf::io::FilterInputStream::markSupported () const` [inline, virtual]

Determines if this input stream supports the mark and reset methods.

Whether or not mark and reset are supported is an invariant property of a particular input stream instance.

Returns

true if this stream instance supports marks

Implements `decaf::io::InputStream` (p. 1631).

Reimplemented in `decaf::io::BufferedInputStream` (p. 750).

References `DECAF_CATCHALL_NOTHROW`.

6.310.3.7 `virtual void decaf::io::FilterInputStream::notify ()
 throw (decaf::lang::exceptions::RuntimeException,
 decaf::lang::exceptions::IllegalMonitorStateException) [inline,
 virtual]`

Signals a waiter on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

RuntimeException if an error occurs while notifying one of the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 2933).

6.310.3.8 `virtual void decaf::io::FilterInputStream::notifyAll ()
 throw (decaf::lang::exceptions::RuntimeException,
 decaf::lang::exceptions::IllegalMonitorStateException) [inline,
 virtual]`

Signals the waiters on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

RuntimeException if an error occurs while notifying the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 2934).

6.310.3.9 `virtual int decaf::io::FilterInputStream::read () throw (IOException)
 [inline, virtual]`

Reads the next byte of data from this input stream.

The value byte is returned as an unsigned char in the range 0 to 255. If no byte is available because the end of the stream has been reached, the value -1 is returned. This method blocks until input data is available, the end of the stream is detected, or an exception is thrown. This method simply performs `in.read()` and returns the result.

Returns

The next byte.

Exceptions

IOException (p. 1707) thrown if an error occurs.

Implements **decaf::io::InputStream** (p. 1632).

Reimplemented in **activemq::io::LoggingInputStream** (p. 1918), **decaf::io::BufferedInputStream** (p. 751), and **decaf::io::DataInputStream** (p. 1293).

References **DECAF_CATCH_RETHROW**, and **DECAF_CATCHALL_THROW**.

Referenced by **decaf::io::DataInputStream::read()**.

6.310.3.10 **virtual int decaf::io::FilterInputStream::read (unsigned char * *buffer*, std::size_t *offset*, std::size_t *bufferSize*) throw (IOException, lang::exceptions::NullPointerException) [inline, virtual]**

Reads up to len bytes of data from this input stream into an array of bytes.

This method blocks until some input is available. This method simply performs **in.read(b, len)** and returns the result.

Parameters

buffer (out) the target buffer.

offset the position to start reading in the passed buffer.

bufferSize the size of the output buffer.

Returns

The number of bytes read or -1 if EOF is detected

Exceptions

IOException (p. 1707) thrown if an error occurs.

NullPointerException

Implements **decaf::io::InputStream** (p. 1632).

Reimplemented in **activemq::io::LoggingInputStream** (p. 1918), **decaf::io::BufferedInputStream** (p. 750), and **decaf::io::DataInputStream** (p. 1294).

References **DECAF_CATCH_RETHROW**, and **DECAF_CATCHALL_THROW**.

6.310.3.11 **virtual void decaf::io::FilterInputStream::reset () throw (IOException) [inline, virtual]**

Repositions this stream to the position at the time the mark method was last called on this input stream.

If the method **markSupported** returns true, then: * If the method **mark** has not been called since the stream was created, or the number of bytes read from the stream since **mark** was last called is larger than the argument to **mark** at that last call, then an **IOException** (p. 1707) might be thrown. * If such an **IOException** (p. 1707) is not thrown, then the stream is reset to a state such that all the bytes read since the most recent call to **mark** (or since the start of the file, if **mark** has not been called) will be resupplied to subsequent callers of the **read** method, followed by any bytes that otherwise would have been the next input data as of the time of the call to **reset**. If the method **markSupported** returns false, then: * The call to **reset** may throw an **IOException** (p. 1707). * If an **IOException** (p. 1707) is not thrown, then the stream is reset to a fixed state that depends on the particular type of the input stream and how it was created. The bytes that will be supplied to subsequent callers of the **read** method depend on the particular type of the input stream.

Exceptions***IOException*** (p. 1707)Implements **decaf::io::InputStream** (p. 1633).Reimplemented in **decaf::io::BufferedInputStream** (p. 751).

References DECAF_CATCH_RETHROW, and DECAF_CATCHALL_THROW.

6.310.3.12 `virtual std::size_t decaf::io::FilterInputStream::skip
 (std::size_t num) throw (io::IOException,
 lang::exceptions::UnsupportedOperationException) [inline, virtual]`

Skips over and discards n bytes of data from this input stream.

The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before n bytes have been skipped is only one possibility. The actual number of bytes skipped is returned. If n is negative, no bytes are skipped.

The skip method of **InputStream** (p. 1630) creates a byte array and then repeatedly reads into it until n bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.

Parameters*num* - the number of bytes to skip**Returns**

total butes skipped

Exceptions***IOException*** (p. 1707) if an error occursImplements **decaf::io::InputStream** (p. 1633).Reimplemented in **decaf::io::BufferedInputStream** (p. 751), and **decaf::io::DataInputStream** (p. 1300).

References DECAF_CATCH_RETHROW, and DECAF_CATCHALL_THROW.

6.310.3.13 `virtual bool decaf::io::FilterInputStream::tryLock () throw (decaf::lang::exceptions::RuntimeException) [inline, virtual]`

Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

Returns

true if the lock was acquired, false if it is already held by another thread.

Exceptions***RuntimeException*** if an error occurs while locking the object.Implements **decaf::util::concurrent::Synchronizable** (p. 2935).

6.310.3.14 virtual void decaf::io::FilterInputStream::unlock () throw (decaf::lang::exceptions::RuntimeException) [inline, virtual]

Unlocks the object.

Exceptions

RuntimeException if an error occurs while unlocking the object.

Implements decaf::util::concurrent::Synchronizable (p. 2936).

6.310.3.15 virtual void decaf::io::FilterInputStream::wait (long long *millisecs*, int *nanos*) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException) [inline, virtual]

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

Parameters

millisecs the time in milliseconds to wait, or WAIT_INFINITE

nanos additional time in nanoseconds with a range of 0-999999

Exceptions

IllegalArgumentException if an error occurs or the nanos argument is not in the range of [0-999999]

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements decaf::util::concurrent::Synchronizable (p. 2940).

6.310.3.16 virtual void decaf::io::FilterInputStream::wait (long long *millisecs*) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException) [inline, virtual]

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval.

Parameters

millisecs the time in milliseconds to wait, or WAIT_INFINITE

Exceptions

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements `decaf::util::concurrent::Synchronizable` (p. 2938).

```
6.310.3.17 virtual void decaf::io::FilterInputStream::wait (    )
            throw ( decaf::lang::exceptions::RuntimeException,
                    decaf::lang::exceptions::IllegalMonitorStateException,
                    decaf::lang::exceptions::InterruptedException ) [inline, virtual]
```

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling.

Exceptions

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements `decaf::util::concurrent::Synchronizable` (p. 2937).

6.310.4 Field Documentation

6.310.4.1 `volatile bool decaf::io::FilterInputStream::closed` [protected]

6.310.4.2 `InputStream* decaf::io::FilterInputStream::inputStream` [protected]

6.310.4.3 `util::concurrent::Mutex decaf::io::FilterInputStream::mutex` [protected]

6.310.4.4 `bool decaf::io::FilterInputStream::own` [protected]

The documentation for this class was generated from the following file:

- `src/main/decaf/io/FilterInputStream.h`

6.311 decaf::io::FilterOutputStream Class Reference

This class is the superclass of all classes that filter output streams.

```
#include <src/main/decaf/io/FilterOutputStream.h>
```

Inheritance diagram for `decaf::io::FilterOutputStream`:

Public Member Functions

- **FilterOutputStream** (**OutputStream** *outputStream, bool own=false)
Constructor, creates a wrapped output stream.
- virtual ~**FilterOutputStream** ()
- virtual void **write** (unsigned char c) throw (**IOException**)
Writes a single byte to the output stream.
- virtual void **write** (const std::vector< unsigned char > &buffer) throw (**IOException**)
Writes an array of bytes to the output stream.
- virtual void **write** (const unsigned char *buffer, std::size_t offset DECAF_UNUSED, std::size_t len) throw (**IOException**, lang::exceptions::NullPointerException)
Writes an array of bytes to the output stream.
- virtual void **flush** () throw (**IOException**)
Flushes any pending writes in this output stream.
- virtual void **close** () throw (io::IOException)
Close the Stream.
- virtual void **lock** () throw (decaf::lang::exceptions::RuntimeException)
Locks the object.
- virtual bool **tryLock** () throw (decaf::lang::exceptions::RuntimeException)
Attempts to Lock the object, if the lock is already held by another thread than this method returns false.
- virtual void **unlock** () throw (decaf::lang::exceptions::RuntimeException)
Unlocks the object.
- virtual void **wait** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (long long millisecs) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (long long millisecs, int nanos) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **notify** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)
Signals a waiter on this object that it can now wake up and continue.

- virtual void **notifyAll** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)

Signals the waiters on this object that it can now wake up and continue.

Protected Member Functions

- virtual bool **isClosed** () const

Protected Attributes

- **OutputStream** * **outputStream**
- **util::concurrent::Mutex** **mutex**
- bool **own**
- volatile bool **closed**

6.311.1 Detailed Description

This class is the superclass of all classes that filter output streams. These streams sit on top of an already existing output stream (the underlying output stream) which it uses as its basic sink of data, but possibly transforming the data along the way or providing additional functionality.

The class **FilterOutputStream** (p. 1536) itself simply overrides all methods of **OutputStream** (p. 2316) with versions that pass all requests to the underlying output stream. Subclasses of **FilterOutputStream** (p. 1536) may further override some of these methods as well as provide additional methods and fields.

Due to the lack of garbage collection in C++ a design decision was made to add a boolean parameter to the constructor indicating if the wrapped **InputStream** (p. 1630) is owned by this object. That way creation of the underlying stream can occur in a Java like way. Ex:

DataOutputStream (p. 1300) `os = new DataOutputStream (p. 1300)(new OutputStream(), true)`

6.311.2 Constructor & Destructor Documentation

6.311.2.1 decaf::io::FilterOutputStream::FilterOutputStream (**OutputStream** * *outputStream*, bool *own* = *false*) [inline]

Constructor, creates a wrapped output stream.

Parameters

outputStream the **OutputStream** (p. 2316) to wrap

own If true, this object will control the lifetime of the output stream that it encapsulates.

6.311.2.2 virtual decaf::io::FilterOutputStream::~~FilterOutputStream () [inline, virtual]

References DECAF_CATCH_NOTHROW, and DECAF_CATCHALL_NOTHROW.

6.311.3 Member Function Documentation

6.311.3.1 virtual void decaf::io::FilterOutputStream::close () throw (io::IOException) [inline, virtual]

Close the Stream.

The close method of **FilterOutputStream** (p.1536) calls its flush method, and then calls the close method of its underlying output stream, it then destroys the output stream if it is the owner.

Exceptions

IOException

Reimplemented in **decaf::io::BufferedOutputStream** (p. 753).

References DECAF_CATCH_RETHROW, and DECAF_CATCHALL_THROW.

6.311.3.2 virtual void decaf::io::FilterOutputStream::flush () throw (IOException) [inline, virtual]

Flushes any pending writes in this output stream.

The flush method of **FilterOutputStream** (p.1536) calls the flush method of its underlying output stream

Exceptions

IOException (p. 1707)

Implements **decaf::io::OutputStream** (p. 2317).

Reimplemented in **decaf::io::BufferedOutputStream** (p. 753).

References DECAF_CATCH_RETHROW, and DECAF_CATCHALL_THROW.

6.311.3.3 virtual bool decaf::io::FilterOutputStream::isClosed () const [inline, protected, virtual]

Returns

true if this stream has been closed.

6.311.3.4 virtual void decaf::io::FilterOutputStream::lock () throw (decaf::lang::exceptions::RuntimeException) [inline, virtual]

Locks the object.

Exceptions

RuntimeException if an error occurs while locking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2932).

6.311.3.5 `virtual void decaf::io::FilterOutputStream::notify ()
throw (decaf::lang::exceptions::RuntimeException,
decaf::lang::exceptions::IllegalMonitorStateException) [inline,
virtual]`

Signals a waiter on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

RuntimeException if an error occurs while notifying one of the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 2933).

6.311.3.6 `virtual void decaf::io::FilterOutputStream::notifyAll
() throw (decaf::lang::exceptions::RuntimeException,
decaf::lang::exceptions::IllegalMonitorStateException) [inline,
virtual]`

Signals the waiters on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

RuntimeException if an error occurs while notifying the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 2934).

6.311.3.7 `virtual bool decaf::io::FilterOutputStream::tryLock () throw (decaf::lang::exceptions::RuntimeException) [inline, virtual]`

Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

Returns

true if the lock was acquired, false if it is already held by another thread.

Exceptions

RuntimeException if an error occurs while locking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2935).

6.311.3.8 `virtual void decaf::io::FilterOutputStream::unlock () throw (decaf::lang::exceptions::RuntimeException) [inline, virtual]`

Unlocks the object.

Exceptions

RuntimeException if an error occurs while unlocking the object.

Implements `decaf::util::concurrent::Synchronizable` (p. 2936).

6.311.3.9 `virtual void decaf::io::FilterOutputStream::wait ()
throw (decaf::lang::exceptions::RuntimeException,
decaf::lang::exceptions::IllegalMonitorStateException,
decaf::lang::exceptions::InterruptedException) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling.

Exceptions

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements `decaf::util::concurrent::Synchronizable` (p. 2937).

6.311.3.10 `virtual void decaf::io::FilterOutputStream::wait (long long
millisecs) throw (decaf::lang::exceptions::RuntimeException,
decaf::lang::exceptions::IllegalMonitorStateException,
decaf::lang::exceptions::InterruptedException) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval.

Parameters

millisecs the time in milliseconds to wait, or WAIT_INFINITE

Exceptions

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements `decaf::util::concurrent::Synchronizable` (p. 2938).

6.311.3.11 `virtual void decaf::io::FilterOutputStream::wait (long long millisecs,
int nanos) throw (decaf::lang::exceptions::RuntimeException,
decaf::lang::exceptions::IllegalArgumentException,
decaf::lang::exceptions::IllegalMonitorStateException,
decaf::lang::exceptions::InterruptedException) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

Parameters

milliseconds the time in milliseconds to wait, or WAIT_INFINITE

nanos additional time in nanoseconds with a range of 0-999999

Exceptions

IllegalArgumentException if an error occurs or the nanos argument is not in the range of [0-999999]

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2940).

6.311.3.12 `virtual void decaf::io::FilterOutputStream::write (const std::vector< unsigned char > & buffer) throw (IOException) [inline, virtual]`

Writes an array of bytes to the output stream.

Parameters

buffer The bytes to write.

Exceptions

IOException (p. 1707) thrown if an error occurs.

Implements **decaf::io::OutputStream** (p. 2317).

Reimplemented in **decaf::io::BufferedOutputStream** (p. 754), and **decaf::io::DataOutputStream** (p. 1303).

References DECAF_CATCH_RETHROW, and DECAF_CATCHALL_THROW.

6.311.3.13 `virtual void decaf::io::FilterOutputStream::write (unsigned char c) throw (IOException) [inline, virtual]`

Writes a single byte to the output stream.

The write method of **FilterOutputStream** (p. 1536) calls the write method of its underlying output stream, that is, it performs out.write(b).

Parameters

c the byte.

Exceptions

IOException (p. 1707) thrown if an error occurs.

Implements **decaf::io::OutputStream** (p. 2318).

Reimplemented in **activemq::io::LoggingOutputStream** (p. 1919), **decaf::io::BufferedOutputStream** (p. 754), and **decaf::io::DataOutputStream** (p. 1302).

References **DECAF_CATCH_RETHROW**, and **DECAF_CATCHALL_THROW**.

6.311.3.14 `virtual void decaf::io::FilterOutputStream::write (const unsigned char * buffer, std::size_t offset DECAF_UNUSED, std::size_t len) throw (IOException, lang::exceptions::NullPointerException)`
[inline, virtual]

Writes an array of bytes to the output stream.

The write method of **FilterOutputStream** (p. 1536) calls the write method of one argument on each byte to output.

Parameters

buffer The array of bytes to write.

offset, the position to start writing in buffer.

len The number of bytes from the buffer to be written.

Exceptions

IOException (p. 1707) thrown if an error occurs.

NullPointerException thrown if buffer is Null.

Implements **decaf::io::OutputStream** (p. 2317).

Reimplemented in **activemq::io::LoggingOutputStream** (p. 1920), **decaf::io::BufferedOutputStream** (p. 754), and **decaf::io::DataOutputStream** (p. 1303).

References **DECAF_CATCH_RETHROW**, and **DECAF_CATCHALL_THROW**.

6.311.4 Field Documentation

6.311.4.1 `volatile bool decaf::io::FilterOutputStream::closed` [protected]

6.311.4.2 `util::concurrent::Mutex decaf::io::FilterOutputStream::mutex`
[protected]

6.311.4.3 `OutputStream* decaf::io::FilterOutputStream::outputStream`
[protected]

6.311.4.4 `bool decaf::io::FilterOutputStream::own` [protected]

The documentation for this class was generated from the following file:

- `src/main/decaf/io/FilterOutputStream.h`

6.312 decaf::lang::Float Class Reference

```
#include <src/main/decaf/lang/Float.h>
```

Inheritance diagram for decaf::lang::Float:

Public Member Functions

- **Float** (float value)
- **Float** (double value)
- **Float** (const std::string &value) throw (exceptions::NumberFormatException)
- virtual ~**Float** ()
- virtual int **compareTo** (const **Float** &f) const
*Compares this **Float** (p. 1544) instance with another.*
- bool **equals** (const **Float** &f) const
- virtual bool **operator==** (const **Float** &f) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const **Float** &f) const
Compares this object to another and returns true if this object is considered to be less than the one passed.
- virtual int **compareTo** (const float &f) const
*Compares this **Float** (p. 1544) instance with another.*
- bool **equals** (const float &f) const
- virtual bool **operator==** (const float &f) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const float &f) const
Compares this object to another and returns true if this object is considered to be less than the one passed.
- std::string **toString** () const
- virtual double **doubleValue** () const
Answers the double value which the receiver represents.
- virtual float **floatValue** () const
Answers the float value which the receiver represents.
- virtual unsigned char **byteValue** () const
Answers the byte value which the receiver represents.
- virtual short **shortValue** () const
Answers the short value which the receiver represents.
- virtual int **intValue** () const

Answers the int value which the receiver represents.

- virtual long long **longValue** () const

Answers the long value which the receiver represents.

- bool **isInfinite** () const
- bool **isNaN** () const

Static Public Member Functions

- static int **compare** (float f1, float f2)

Compares the two specified double values.

- static int **floatToIntBits** (float value)

Returns a representation of the specified floating-point value according to the IEEE 754 floating-point "single format" bit layout.

- static int **floatToRawIntBits** (float value)

Returns a representation of the specified floating-point value according to the IEEE 754 floating-point "single format" bit layout, preserving Not-a-Number (NaN) values.

- static float **intBitsToFloat** (int bits)

Returns the float value corresponding to a given bit representation.

- static bool **isInfinite** (float value)

- static bool **isNaN** (float value)

- static float **parseFloat** (const std::string &value) throw (exceptions::NumberFormatException)

*Returns a new float initialized to the value represented by the specified string, as performed by the valueOf method of class **Float** (p. 1544).*

- static std::string **toHexString** (float value)

Returns a hexadecimal string representation of the float argument.

- static std::string **toString** (float value)

Returns a string representation of the float argument.

- static **Float** **valueOf** (float value)

*Returns a **Float** (p. 1544) instance representing the specified float value.*

- static **Float** **valueOf** (const std::string &value) throw (exceptions::NumberFormatException)

*Returns a **Float** (p. 1544) instance that wraps a primitive float which is parsed from the string value passed.*

Static Public Attributes

- static const int **SIZE** = 32
The size in bits of the primitive int type.
- static const float **MAX_VALUE**
The maximum value that the primitive type can hold.
- static const float **MIN_VALUE**
The minimum value that the primitive type can hold.
- static const float **NaN**
*Constant for the Not a **Number** (p. 2282) Value.*
- static const float **POSITIVE_INFINITY**
Constant for Positive Infinity.
- static const float **NEGATIVE_INFINITY**
Constant for Negative Infinity.

6.312.1 Constructor & Destructor Documentation

6.312.1.1 `decaf::lang::Float::Float (float value)`

Parameters

value - the primitive type to wrap

6.312.1.2 `decaf::lang::Float::Float (double value)`

Parameters

value - the primitive type to wrap

6.312.1.3 `decaf::lang::Float::Float (const std::string & value) throw (exceptions::NumberFormatException)`

Parameters

value - the string to convert to a primitive type to wrap

6.312.1.4 `virtual decaf::lang::Float::~~Float () [inline, virtual]`

6.312.2 Member Function Documentation

6.312.2.1 `virtual unsigned char decaf::lang::Float::byteValue () const [inline, virtual]`

Answers the byte value which the receiver represents.

Returns

byte the value of the receiver.

6.312.2.2 static int decaf::lang::Float::compare (float *f1*, float *f2*) [static]

Compares the two specified double values.

The sign of the integer value returned is the same as that of the integer that would be returned by the call: `Float(f1).compareTo(Float(f2))`

Parameters

f1 - the first double to compare

f2 - the second double to compare

Returns

the value 0 if *d1* is numerically equal to *f2*; a value less than 0 if *f1* is numerically less than *f2*; and a value greater than 0 if *f1* is numerically greater than *f2*.

6.312.2.3 virtual int decaf::lang::Float::compareTo (const float & *f*) const [virtual]

Compares this **Float** (p. 1544) instance with another.

Parameters

f - the **Float** (p. 1544) instance to be compared

Returns

zero if this object represents the same integer value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

Implements **decaf::lang::Comparable**< float > (p. 1010).

6.312.2.4 virtual int decaf::lang::Float::compareTo (const Float & *f*) const [virtual]

Compares this **Float** (p. 1544) instance with another.

Parameters

f - the **Float** (p. 1544) instance to be compared

Returns

zero if this object represents the same integer value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

6.312.2.5 `virtual double decaf::lang::Float::doubleValue () const [inline, virtual]`

Answers the double value which the receiver represents.

Returns

double the value of the receiver.

6.312.2.6 `bool decaf::lang::Float::equals (const Float & f) const [inline]`

Parameters

f - the **Float** (p.1544) object to compare against.

Returns

true if the two **Float** (p.1544) Objects have the same value.

6.312.2.7 `bool decaf::lang::Float::equals (const float & f) const [inline, virtual]`

Parameters

f - the **Float** (p.1544) object to compare against.

Returns

true if the two **Float** (p.1544) Objects have the same value.

Implements **decaf::lang::Comparable**< **float** > (p.1010).

6.312.2.8 `static int decaf::lang::Float::floatToIntBits (float value) [static]`

Returns a representation of the specified floating-point value according to the IEEE 754 floating-point "single format" bit layout.

Bit 31 (the bit that is selected by the mask 0x80000000) represents the sign of the floating-point number. Bits 30-23 (the bits that are selected by the mask 0x7f800000) represent the exponent. Bits 22-0 (the bits that are selected by the mask 0x007fffff) represent the significand (sometimes called the mantissa) of the floating-point number.

If the argument is positive infinity, the result is 0x7f800000. If the argument is negative infinity, the result is 0xff800000. If the argument is NaN, the result is 0x7fc00000.

In all cases, the result is an integer that, when given to the **intBitsToFloat(int)** (p.1549) method, will produce a floating-point value the same as the argument to floatToIntBits (except all NaN values are collapsed to a single "canonical" NaN value).

Parameters

value - the float to convert to int bits

Returns

the int that holds the float's value

6.312.2.9 static int decaf::lang::Float::floatToRawIntBits (float *value*) [static]

Returns a representation of the specified floating-point value according to the IEEE 754 floating-point "single format" bit layout, preserving Not-a-Number (NaN) values.

Bit 31 (the bit that is selected by the mask 0x80000000) represents the sign of the floating-point number. Bits 30-23 (the bits that are selected by the mask 0x7f800000) represent the exponent. Bits 22-0 (the bits that are selected by the mask 0x007fffff) represent the significand (sometimes called the mantissa) of the floating-point number.

If the argument is positive infinity, the result is 0x7f800000. If the argument is negative infinity, the result is 0xff800000. If the argument is NaN, the result is the integer representing the actual NaN value. Unlike the floatToIntBits method, intToRawIntBits does not collapse all the bit patterns encoding a NaN to a single "canonical" NaN value.

In all cases, the result is an integer that, when given to the **intBitsToFloat(int)** (p. 1549) method, will produce a floating-point value the same as the argument to floatToRawIntBits.

Parameters

value The float to convert to a raw int.

Returns

the raw int value of the float

6.312.2.10 virtual float decaf::lang::Float::floatValue () const [inline, virtual]

Answers the float value which the receiver represents.

Returns

float the value of the receiver.

6.312.2.11 static float decaf::lang::Float::intBitsToFloat (int *bits*) [static]

Returns the float value corresponding to a given bit representation.

The argument is considered to be a representation of a floating-point value according to the IEEE 754 floating-point "single format" bit layout.

If the argument is 0x7f800000, the result is positive infinity. If the argument is 0xff800000, the result is negative infinity. If the argument is any value in the range 0x7f800001 through 0x7fffffff or in the range 0xff800001 through 0xffffffff, the result is a NaN. No IEEE 754 floating-point operation provided by C++ can distinguish between two NaN values of the same type with different bit patterns. Distinct values of NaN are only distinguishable by use of the **Float::floatToRawIntBits** (p. 1549) method.

Parameters

bits - the bits of the float encoded as a float

Returns

a new float created from the int bits.

6.312.2.12 `virtual int decaf::lang::Float::intValue () const [inline, virtual]`

Answers the int value which the receiver represents.

Returns

int the value of the receiver.

6.312.2.13 `bool decaf::lang::Float::isInfinite () const`

Returns

true if the float is equal to positive infinity.

6.312.2.14 `static bool decaf::lang::Float::isInfinite (float value) [static]`

Parameters

value - The float to check.

Returns

true if the float is equal to infinity.

6.312.2.15 `bool decaf::lang::Float::isNaN () const`

Returns

true if the float is equal to NaN.

6.312.2.16 `static bool decaf::lang::Float::isNaN (float value) [static]`

Parameters

value - The float to check.

Returns

true if the float is equal to NaN.

6.312.2.17 `virtual long long decaf::lang::Float::longValue () const [inline, virtual]`

Answers the long value which the receiver represents.

Returns

long the value of the receiver.

6.312.2.18 `virtual bool decaf::lang::Float::operator< (const float & f) const`
[inline, virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

Parameters

f - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

Implements `decaf::lang::Comparable< float >` (p.1011).

6.312.2.19 `virtual bool decaf::lang::Float::operator< (const Float & f) const`
[inline, virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

Parameters

f - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

6.312.2.20 `virtual bool decaf::lang::Float::operator== (const Float & f) const`
[inline, virtual]

Compares equality between this object and the one passed.

Parameters

f - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

6.312.2.21 `virtual bool decaf::lang::Float::operator== (const float & f) const`
[inline, virtual]

Compares equality between this object and the one passed.

Parameters

f - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable**< **float** > (p. 1011).

6.312.2.22 **static float decaf::lang::Float::parseFloat (const std::string & value)
 throw (exceptions::NumberFormatException) [static]**

Returns a new float initialized to the value represented by the specified string, as performed by the **valueOf** method of class **Float** (p. 1544).

Parameters

value - the string to parse

Returns

a float parsed from the string

Exceptions

NumberFormatException

6.312.2.23 **virtual short decaf::lang::Float::shortValue () const [inline,
 virtual]**

Answers the short value which the receiver represents.

Returns

short the value of the receiver.

6.312.2.24 **static std::string decaf::lang::Float::toHexString (float value)
 [static]**

Returns a hexadecimal string representation of the float argument.

All characters mentioned below are ASCII characters.

* If the argument is NaN, the result is the string "NaN". * Otherwise, the result is a string that represents the sign and magnitude (absolute value) of the argument. If the sign is negative, the first character of the result is '-'; if the sign is positive, no sign character appears in the result. As for the magnitude m: o If m is infinity, it is represented by the string "Infinity"; thus, positive infinity produces the result "Infinity" and negative infinity produces the result "-Infinity". o If m is zero, it is represented by the string "0x0.0p0"; thus, negative zero produces the result "-0x0.0p0" and positive zero produces the result "0x0.0p0". o If m is a float value with a normalized representation, substrings are used to represent the significand and exponent fields. The significand is represented by the characters "0x1." followed by a lowercase hexadecimal representation of the rest of the significand as a fraction. Trailing zeros in the hexadecimal representation are removed unless all the digits are zero, in which case a single zero is used. Next, the exponent is represented by "p" followed by a decimal string of the unbiased exponent as if produced by a call to **Integer.toString** (p. 1664) on the exponent value. o If m is a float value

with a subnormal representation, the significand is represented by the characters "0x0." followed by a hexadecimal representation of the rest of the significand as a fraction. Trailing zeros in the hexadecimal representation are removed. Next, the exponent is represented by "p-126". Note that there must be at least one nonzero digit in a subnormal significand.

Parameters

value - The float to convert to a string

Returns

the Hex formatted float string.

6.312.2.25 std::string decaf::lang::Float::toString () const

Returns

this **Float** (p. 1544) Object as a String Representation

6.312.2.26 static std::string decaf::lang::Float::toString (float value) [static]

Returns a string representation of the float argument.

All characters mentioned below are ASCII characters.

If the argument is NaN, the result is the string "NaN". Otherwise, the result is a string that represents the sign and magnitude (absolute value) of the argument. If the sign is negative, the first character of the result is '-'; if the sign is positive, no sign character appears in the result. As for the magnitude m: o If m is infinity, it is represented by the characters "Infinity"; thus, positive infinity produces the result "Infinity" and negative infinity produces the result "-Infinity". o If m is zero, it is represented by the characters "0.0"; thus, negative zero produces the result "-0.0" and positive zero produces the result "0.0". o If m is greater than or equal to 10⁻³ but less than 10⁷, then it is represented as the integer part of m, in decimal form with no leading zeroes, followed by '.', followed by one or more decimal digits representing the fractional part of m. o If m is less than 10⁻³ or greater than or equal to 10⁷, then it is represented in so-called "computerized scientific notation." Let n be the unique integer such that 10ⁿ ≤ m < 10ⁿ⁺¹; then let a be the mathematically exact quotient of m and 10ⁿ so that 1 ≤ a < 10. The magnitude is then represented as the integer part of a, as a single decimal digit, followed by '.', followed by decimal digits representing the fractional part of a, followed by the letter 'E', followed by a representation of n as a decimal integer, as produced by the method **Integer.toString(int)** (p. 1665).

Parameters

value - The float to convert to a string

Returns

the formatted float string.

6.312.2.27 static Float decaf::lang::Float::valueOf (const std::string & value) throw (exceptions::NumberFormatException) [static]

Returns a **Float** (p. 1544) instance that wraps a primitive float which is parsed from the string value passed.

Parameters

value - the string to parse

Returns

a new **Float** (p. 1544) instance wrapping the float parsed from value

Exceptions

NumberFormatException on error.

6.312.2.28 `static Float decaf::lang::Float::valueOf (float value) [static]`

Returns a **Float** (p. 1544) instance representing the specified float value.

Parameters

value - float to wrap

Returns

new **Float** (p. 1544) instance wrapping the primitive value

6.312.3 Field Documentation

6.312.3.1 `const float decaf::lang::Float::MAX_VALUE [static]`

The maximum value that the primitive type can hold.

6.312.3.2 `const float decaf::lang::Float::MIN_VALUE [static]`

The minimum value that the primitive type can hold.

6.312.3.3 `const float decaf::lang::Float::NaN [static]`

Constant for the Not a **Number** (p. 2282) Value.

6.312.3.4 `const float decaf::lang::Float::NEGATIVE_INFINITY [static]`

Constant for Negative Infinity.

6.312.3.5 `const float decaf::lang::Float::POSITIVE_INFINITY [static]`

Constant for Positive Infinity.

6.312.3.6 `const int decaf::lang::Float::SIZE = 32 [static]`

The size in bits of the primitive int type.

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Float.h`

6.313 decaf::internal::nio::FloatArrayBuffer Class Reference

```
#include <src/main/decaf/internal/nio/FloatArrayBuffer.h>
```

Inheritance diagram for decaf::internal::nio::FloatArrayBuffer:

Public Member Functions

- **FloatArrayBuffer** (std::size_t capacity, bool readOnly=false)
*Creates a **FloatArrayBuffer** (p. 1555) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*
- **FloatArrayBuffer** (float *array, std::size_t offset, std::size_t capacity, bool readOnly=false) throw (decaf::lang::exceptions::NullPointerException)
*Creates a **FloatArrayBuffer** (p. 1555) object that wraps the given array.*
- **FloatArrayBuffer** (ByteArrayPerspective &array, std::size_t offset, std::size_t capacity, bool readOnly=false) throw (decaf::lang::exceptions::IndexOutOfBoundsException)
*Creates a byte buffer that wraps the passed **ByteArrayPerspective** (p. 843) and start at the given offset.*
- **FloatArrayBuffer** (const FloatArrayBuffer &other)
*Create a **FloatArrayBuffer** (p. 1555) that mirrors this one, meaning it shares a reference to this buffers **ByteArrayPerspective** (p. 843) and when changes are made to that data it is reflected in both.*
- virtual ~FloatArrayBuffer ()
- virtual float * **array** () throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException)
Returns the float array that backs this buffer (optional operation).
- virtual std::size_t **arrayOffset** () throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException)
Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).
- virtual FloatBuffer * **asReadOnlyBuffer** () const
Creates a new, read-only float buffer that shares this buffer's content.
- virtual FloatBuffer & **compact** () throw (decaf::nio::ReadOnlyBufferException)
Compacts this buffer.
- virtual FloatBuffer * **duplicate** ()
Creates a new float buffer that shares this buffer's content.
- virtual float **get** () throw (decaf::nio::BufferUnderflowException)
Relative get method.
- virtual float **get** (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)

Absolute get method.

- virtual bool **hasArray** () const
Tells whether or not this buffer is backed by an accessible float array.
- virtual bool **isReadOnly** () const
Tells whether or not this buffer is read-only.
- virtual FloatBuffer & **put** (float value) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException)
Writes the given doubles into this buffer at the current position, and then increments the position.
- virtual FloatBuffer & **put** (std::size_t index, float value) throw (lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException)
Writes the given doubles into this buffer at the given index.
- virtual FloatBuffer * **slice** () const
Creates a new FloatBuffer whose content is a shared subsequence of this buffer's content.

Protected Member Functions

- virtual void **setReadOnly** (bool value)
*Sets this **ByteBuffer** (p. 806) as Read-Only.*

6.313.1 Constructor & Destructor Documentation

6.313.1.1 decaf::internal::nio::FloatArrayBuffer::FloatArrayBuffer (std::size_t capacity, bool readOnly = false)

Creates a **FloatArrayBuffer** (p. 1555) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.

The array is initially created with all elements initialized to zero.

Parameters

capacity - size of the array, this is the limit we read and write to.

readOnly - should this buffer be read-only, default as false

6.313.1.2 decaf::internal::nio::FloatArrayBuffer::FloatArrayBuffer (float * array, std::size_t offset, std::size_t capacity, bool readOnly = false) throw (decaf::lang::exceptions::NullPointerException)

Creates a **FloatArrayBuffer** (p. 1555) object that wraps the given array.

If the own flag is set then it will delete this array when this object is deleted.

Parameters

array - array to wrap
offset - the position that is this buffers start pos.
capacity - size of the array, this is the limit we read and write to.
readOnly - should this buffer be read-only, default as false

Exceptions

NullPointerException if buffer is NULL

6.313.1.3 decaf::internal::nio::FloatArrayBuffer::FloatArrayBuffer (ByteArrayPerspective & array, std::size_t offset, std::size_t capacity, bool readOnly = false) throw (decaf::lang::exceptions::IndexOutOfBoundsException)

Creates a byte buffer that wraps the passed **ByteArrayPerspective** (p. 843) and start at the given offset.

The capacity and limit of the new **FloatArrayBuffer** (p. 1555) will be that of the remaining capacity of the passed buffer.

Parameters

array - the **ByteArrayPerspective** (p. 843) to wrap
offset - the offset into array where the buffer starts
capacity - the length of the array we are wrapping or limit.
readOnly - is this a readOnly buffer.

Exceptions

IndexOutOfBoundsException if offset is greater than array capacity.

6.313.1.4 decaf::internal::nio::FloatArrayBuffer::FloatArrayBuffer (const FloatArrayBuffer & other)

Create a **FloatArrayBuffer** (p. 1555) that mirrors this one, meaning it shares a reference to this buffers **ByteArrayPerspective** (p. 843) and when changes are made to that data it is reflected in both.

Parameters

other - the **FloatArrayBuffer** (p. 1555) this one is to mirror.

6.313.1.5 virtual decaf::internal::nio::FloatArrayBuffer::~FloatArrayBuffer () [virtual]

6.313.2 Member Function Documentation

6.313.2.1 virtual float* decaf::internal::nio::FloatArrayBuffer::array () throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException) [virtual]

Returns the float array that backs this buffer (optional operation).

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

the array that backs this Buffer

Exceptions

ReadOnlyBufferException if this Buffer is read only.

UnsupportedOperationException if the underlying store has no array.

Implements **decaf::nio::FloatBuffer** (p. 1565).

6.313.2.2 `virtual std::size_t decaf::internal::nio::FloatArrayBuffer::arrayOffset () throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException) [virtual]`

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

The offset into the backing array where index zero starts.

Exceptions

ReadOnlyBufferException if this Buffer is read only.

UnsupportedOperationException if the underlying store has no array.

Implements **decaf::nio::FloatBuffer** (p. 1565).

6.313.2.3 `virtual FloatBuffer* decaf::internal::nio::FloatArrayBuffer::asReadOnlyBuffer () const [virtual]`

Creates a new, read-only float buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the `duplicate` method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns

The new, read-only float buffer which the caller then owns.

Implements **decaf::nio::FloatBuffer** (p. 1565).

**6.313.2.4 virtual FloatBuffer& decaf::internal::nio::FloatArrayBuffer::compact ()
throw (decaf::nio::ReadOnlyBufferException) [virtual]**

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index $p = \text{position}()$ (p. 746) is copied to index zero, the byte at index $p + 1$ is copied to index one, and so forth until the byte at index $\text{limit}()$ (p. 745) - 1 is copied to index $n = \text{limit}()$ (p. 745) - 1 - p . The buffer's position is then set to $n+1$ and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns

a reference to this FloatBuffer

Exceptions

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::FloatBuffer** (p. 1566).

**6.313.2.5 virtual FloatBuffer* decaf::internal::nio::FloatArrayBuffer::duplicate ()
[virtual]**

Creates a new float buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

a new float Buffer which the caller owns.

Implements **decaf::nio::FloatBuffer** (p. 1567).

6.313.2.6 virtual float decaf::internal::nio::FloatArrayBuffer::get () throw (decaf::nio::BufferUnderflowException) [virtual]

Relative get method.

Reads the value at this buffer's current position, and then increments the position.

Returns

the float at the current position

Exceptions

BufferUnderflowException if there no more data to return

Implements **decaf::nio::FloatBuffer** (p. 1568).

6.313.2.7 `virtual float decaf::internal::nio::FloatArrayBuffer::get (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)`
[virtual]

Absolute get method.

Reads the value at the given index.

Parameters

index - the index in the Buffer where the float is to be read

Returns

the float that is located at the given index

Exceptions

IndexOutOfBoundsException - If index is not smaller than the buffer's limit

Implements `decaf::nio::FloatBuffer` (p.1567).

6.313.2.8 `virtual bool decaf::internal::nio::FloatArrayBuffer::hasArray () const`
[inline, virtual]

Tells whether or not this buffer is backed by an accessible float array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Sub-classes should override this method if they do not have a backing array as this class always returns true.

Returns

true if, and only if, this buffer is backed by an array and is not read-only

Implements `decaf::nio::FloatBuffer` (p.1569).

6.313.2.9 `virtual bool decaf::internal::nio::FloatArrayBuffer::isReadOnly () const` [inline, virtual]

Tells whether or not this buffer is read-only.

Returns

true if, and only if, this buffer is read-only

6.313.2.10 `virtual FloatBuffer& decaf::internal::nio::FloatArrayBuffer::put (std::size_t index, float value) throw (lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException)` [virtual]

Writes the given doubles into this buffer at the given index.

Parameters

index - position in the Buffer to write the data

value - the doubles to write.

Returns

a reference to this buffer

Exceptions

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::FloatBuffer** (p.1570).

6.313.2.11 `virtual FloatBuffer& decaf::internal::nio::FloatArrayBuffer::put (float value) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException) [virtual]`

Writes the given doubles into this buffer at the current position, and then increments the position.

Parameters

value - the doubles value to be written

Returns

a reference to this buffer

Exceptions

BufferOverflowException - If this buffer's current position is not smaller than its limit

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::FloatBuffer** (p.1569).

6.313.2.12 `virtual void decaf::internal::nio::FloatArrayBuffer::setReadOnly (bool value) [inline, protected, virtual]`

Sets this **ByteBuffer** (p. 806) as Read-Only.

Parameters

value - true if this buffer is to be read-only.

6.313.2.13 `virtual FloatBuffer* decaf::internal::nio::FloatArrayBuffer::slice () const [virtual]`

Creates a new **FloatBuffer** whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the newly create FloatBuffer which the caller owns.

Implements **decaf::nio::FloatBuffer** (p.1572).

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/nio/FloatArrayBuffer.h`

6.314 decaf::nio::FloatBuffer Class Reference

This class defines four categories of operations upon float buffers:

```
#include <src/main/decaf/nio/FloatBuffer.h>
```

Inheritance diagram for decaf::nio::FloatBuffer:

Public Member Functions

- virtual **~FloatBuffer** ()
- virtual `std::string toString` () const
- virtual `float * array` ()=0 throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException)
Returns the float array that backs this buffer (optional operation).
- virtual `std::size_t arrayOffset` ()=0 throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException)
Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).
- virtual **FloatBuffer * asReadOnlyBuffer** () const =0
Creates a new, read-only float buffer that shares this buffer's content.
- virtual **FloatBuffer & compact** ()=0 throw (ReadOnlyBufferException)
Compacts this buffer.
- virtual **FloatBuffer * duplicate** ()=0
Creates a new float buffer that shares this buffer's content.
- virtual `float get` ()=0 throw (BufferUnderflowException)
Relative get method.
- virtual `float get` (std::size_t index) const =0 throw (lang::exceptions::IndexOutOfBoundsException)
Absolute get method.
- **FloatBuffer & get** (std::vector< float > buffer) throw (BufferUnderflowException)
Relative bulk get method.

- **FloatBuffer** & **get** (float *buffer, std::size_t offset, std::size_t length) throw (BufferUnderflowException, lang::exceptions::NullPointerException)
Relative bulk get method.
- virtual bool **hasArray** () const =0
Tells whether or not this buffer is backed by an accessible float array.
- **FloatBuffer** & **put** (**FloatBuffer** &src) throw (BufferOverflowException, ReadOnlyBufferException, lang::exceptions::IllegalArgumentException)
This method transfers the floats remaining in the given source buffer into this buffer.
- **FloatBuffer** & **put** (const float *buffer, std::size_t offset, std::size_t length) throw (BufferOverflowException, ReadOnlyBufferException, lang::exceptions::NullPointerException)
This method transfers floats into this buffer from the given source array.
- **FloatBuffer** & **put** (std::vector< float > &buffer) throw (BufferOverflowException, ReadOnlyBufferException)
This method transfers the entire content of the given source floats array into this buffer.
- virtual **FloatBuffer** & **put** (float value)=0 throw (BufferOverflowException, ReadOnlyBufferException)
Writes the given floats into this buffer at the current position, and then increments the position.
- virtual **FloatBuffer** & **put** (std::size_t index, float value)=0 throw (lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException)
Writes the given floats into this buffer at the given index.
- virtual **FloatBuffer** * **slice** () const =0
*Creates a new **FloatBuffer** (p.1562) whose content is a shared subsequence of this buffer's content.*
- virtual int **compareTo** (const **FloatBuffer** &value) const
Compares this object with the specified object for order.
- virtual bool **equals** (const **FloatBuffer** &value) const
- virtual bool **operator==** (const **FloatBuffer** &value) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const **FloatBuffer** &value) const
Compares this object to another and returns true if this object is considered to be less than the one passed.

Static Public Member Functions

- static **FloatBuffer** * **allocate** (std::size_t capacity)
Allocates a new Double buffer.
- static **FloatBuffer** * **wrap** (float *array, std::size_t offset, std::size_t length) throw (lang::exceptions::NullPointerException)

*Wraps the passed buffer with a new **FloatBuffer** (p. 1562).*

- static **FloatBuffer** * **wrap** (std::vector< float > &buffer)

*Wraps the passed STL float Vector in a **FloatBuffer** (p. 1562).*

Protected Member Functions

- **FloatBuffer** (std::size_t capacity)

*Creates a **FloatBuffer** (p. 1562) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*

6.314.1 Detailed Description

This class defines four categories of operations upon float buffers: o Absolute and relative get and put methods that read and write single floats; o Relative bulk get methods that transfer contiguous sequences of floats from this buffer into an array; and o Relative bulk put methods that transfer contiguous sequences of floats from a float array or some other float buffer into this buffer o Methods for compacting, duplicating, and slicing a float buffer.

Double buffers can be created either by allocation, which allocates space for the buffer's content, by wrapping an existing float array into a buffer, or by creating a view of an existing byte buffer

Methods in this class that do not otherwise have a value to return are specified to return the buffer upon which they are invoked. This allows method invocations to be chained.

6.314.2 Constructor & Destructor Documentation

6.314.2.1 decaf::nio::FloatBuffer::FloatBuffer (std::size_t capacity) [protected]

Creates a **FloatBuffer** (p. 1562) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.

The array is initially created with all elements initialized to zero.

Parameters

capacity - size and limit of the **Buffer** (p. 741) in doubles

6.314.2.2 virtual decaf::nio::FloatBuffer::~~FloatBuffer () [inline, virtual]

6.314.3 Member Function Documentation

6.314.3.1 static FloatBuffer* decaf::nio::FloatBuffer::allocate (std::size_t capacity) [static]

Allocates a new Double buffer.

The new buffer's position will be zero, its limit will be its capacity, and its mark will be undefined. It will have a backing array, and its array offset will be zero.

Parameters

capacity - The size of the Double buffer in floats

Returns

the **FloatBuffer** (p. 1562) that was allocated, caller owns.

6.314.3.2 `virtual float* decaf::nio::FloatBuffer::array () throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException) [pure virtual]`

Returns the float array that backs this buffer (optional operation).

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

the array that backs this **Buffer** (p. 741)

Exceptions

ReadOnlyBufferException (p. 2520) if this **Buffer** (p. 741) is read only.

UnsupportedOperationException if the underlying store has no array.

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1557).

6.314.3.3 `virtual std::size_t decaf::nio::FloatBuffer::arrayOffset () throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException) [pure virtual]`

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

The offset into the backing array where index zero starts.

Exceptions

ReadOnlyBufferException (p. 2520) if this **Buffer** (p. 741) is read only.

UnsupportedOperationException if the underlying store has no array.

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1558).

6.314.3.4 **virtual FloatBuffer* decaf::nio::FloatBuffer::asReadOnlyBuffer () const** [pure virtual]

Creates a new, read-only float buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns

The new, read-only float buffer which the caller then owns.

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1558).

6.314.3.5 **virtual FloatBuffer& decaf::nio::FloatBuffer::compact () throw (** **ReadOnlyBufferException)** [pure virtual]

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index $p = \text{position}()$ (p. 746) is copied to index zero, the byte at index $p + 1$ is copied to index one, and so forth until the byte at index $\text{limit}()$ (p. 745) - 1 is copied to index $n = \text{limit}()$ (p. 745) - 1 - p . The buffer's position is then set to $n+1$ and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns

a reference to this **FloatBuffer** (p. 1562)

Exceptions

ReadOnlyBufferException (p. 2520) - If this buffer is read-only

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1559).

6.314.3.6 **virtual int decaf::nio::FloatBuffer::compareTo (const FloatBuffer &** **value) const** [virtual]

Compares this object with the specified object for order.

Returns a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

Parameters

value - the Object to be compared.

Returns

a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

6.314.3.7 virtual FloatBuffer* decaf::nio::FloatBuffer::duplicate () [pure virtual]

Creates a new float buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

a new float **Buffer** (p. 741) which the caller owns.

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1559).

6.314.3.8 virtual bool decaf::nio::FloatBuffer::equals (const FloatBuffer & value) const [virtual]
Returns

true if this value is considered equal to the passed value.

6.314.3.9 FloatBuffer& decaf::nio::FloatBuffer::get (std::vector< float > buffer) throw (BufferUnderflowException)

Relative bulk get method.

This method transfers values from this buffer into the given destination vector. An invocation of this method of the form `src.get(a)` behaves in exactly the same way as the invocation. The vector must be sized to the amount of data that is to be read, that is to say, the caller should call `buffer.resize(N)` before calling this get method.

Returns

a reference to this **Buffer** (p. 741)

Exceptions

BufferUnderflowException (p. 774) - If there are fewer than length floats remaining in this buffer

6.314.3.10 virtual float decaf::nio::FloatBuffer::get (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException) [pure virtual]

Absolute get method.

Reads the value at the given index.

Parameters

index - the index in the **Buffer** (p. 741) where the float is to be read

Returns

the float that is located at the given index

Exceptions

IndexOutOfBoundsException - If index is not smaller than the buffer's limit

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1560).

6.314.3.11 **FloatBuffer& decaf::nio::FloatBuffer::get (float * *buffer*, std::size_t *offset*, std::size_t *length*) throw (BufferUnderflowException, lang::exceptions::NullPointerException)**

Relative bulk get method.

This method transfers floats from this buffer into the given destination array. If there are fewer floats remaining in the buffer than are required to satisfy the request, that is, if `length > remaining()` (p. 746), then no bytes are transferred and a **BufferUnderflowException** (p. 774) is thrown.

Otherwise, this method copies `length` floats from this buffer into the given array, starting at the current position of this buffer and at the given offset in the array. The position of this buffer is then incremented by `length`.

Parameters

buffer - pointer to an allocated buffer to fill

offset - position in the buffer to start filling

length - amount of data to put in the passed buffer

Returns

a reference to this **Buffer** (p. 741)

Exceptions

BufferUnderflowException (p. 774) - If there are fewer than `length` floats remaining in this buffer

NullPointerException if the passed buffer is null.

6.314.3.12 **virtual float decaf::nio::FloatBuffer::get () throw (BufferUnderflowException) [pure virtual]**

Relative get method.

Reads the value at this buffer's current position, and then increments the position.

Returns

the float at the current position

Exceptions

BufferUnderflowException (p. 774) if there no more data to return

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1559).

6.314.3.13 virtual bool decaf::nio::FloatBuffer::hasArray () const [pure virtual]

Tells whether or not this buffer is backed by an accessible float array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Sub-classes should override this method if they do not have a backing array as this class always returns true.

Returns

true if, and only if, this buffer is backed by an array and is not read-only

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1560).

6.314.3.14 virtual bool decaf::nio::FloatBuffer::operator< (const FloatBuffer & value) const [virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

Parameters

value - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

6.314.3.15 virtual bool decaf::nio::FloatBuffer::operator== (const FloatBuffer & value) const [virtual]

Compares equality between this object and the one passed.

Parameters

value - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

6.314.3.16 `virtual FloatBuffer& decaf::nio::FloatBuffer::put (float value) throw (BufferOverflowException, ReadOnlyBufferException)` [pure virtual]

Writes the given floats into this buffer at the current position, and then increments the position.

Parameters

value - the floats value to be written

Returns

a reference to this buffer

Exceptions

BufferOverflowException (p. 771) - If this buffer's current position is not smaller than its limit

ReadOnlyBufferException (p. 2520) - If this buffer is read-only

Implemented in `decaf::internal::nio::FloatArrayBuffer` (p. 1561).

6.314.3.17 `virtual FloatBuffer& decaf::nio::FloatBuffer::put (std::size_t index, float value) throw (lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException)` [pure virtual]

Writes the given floats into this buffer at the given index.

Parameters

index - position in the **Buffer** (p. 741) to write the data

value - the floats to write.

Returns

a reference to this buffer

Exceptions

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException (p. 2520) - If this buffer is read-only

Implemented in `decaf::internal::nio::FloatArrayBuffer` (p. 1560).

6.314.3.18 `FloatBuffer& decaf::nio::FloatBuffer::put (const float * buffer, std::size_t offset, std::size_t length) throw (BufferOverflowException, ReadOnlyBufferException, lang::exceptions::NullPointerException)`

This method transfers floats into this buffer from the given source array.

If there are more floats to be copied from the array than remain in this buffer, that is, if `length > remaining()` (p. 746), then no floats are transferred and a **BufferOverflowException** (p. 771) is thrown.

Otherwise, this method copies `length` bytes from the given array into this buffer, starting at the given offset in the array and at the current position of this buffer. The position of this buffer is then incremented by `length`.

Parameters

- buffer*- The array from which floats are to be read
- offset*- The offset within the array of the first float to be read;
- length* - The number of floats to be read from the given array

Returns

a reference to this buffer

Exceptions

- BufferOverflowException* (p. 771) - If there is insufficient space in this buffer
- ReadOnlyBufferException* (p. 2520) - If this buffer is read-only
- NullPointerException* if the passed buffer is null.

6.314.3.19 FloatBuffer& decaf::nio::FloatBuffer::put (std::vector< float > & buffer) throw (BufferOverflowException, ReadOnlyBufferException)

This method transfers the entire content of the given source floats array into this buffer.

This is the same as calling `put(&buffer[0], 0, buffer.size()`

Parameters

- buffer* - The buffer whose contents are copied to this **FloatBuffer** (p. 1562)

Returns

a reference to this buffer

Exceptions

- BufferOverflowException* (p. 771) - If there is insufficient space in this buffer
- ReadOnlyBufferException* (p. 2520) - If this buffer is read-only

6.314.3.20 FloatBuffer& decaf::nio::FloatBuffer::put (FloatBuffer & src) throw (BufferOverflowException, ReadOnlyBufferException, lang::exceptions::IllegalArgumentException)

This method transfers the floats remaining in the given source buffer into this buffer.

If there are more floats remaining in the source buffer than in this buffer, that is, if `src.remaining() > remaining()` (p. 746), then no floats are transferred and a **BufferOverflowException** (p. 771) is thrown.

Otherwise, this method copies `n = src.remaining()` floats from the given buffer into this buffer, starting at each buffer's current position. The positions of both buffers are then incremented by `n`.

Parameters

src - the buffer to take floats from an place in this one.

Returns

a reference to this buffer

Exceptions

BufferOverflowException (p. 771) - If there is insufficient space in this buffer for the remaining floats in the source buffer

IllegalArgumentException - If the source buffer is this buffer

ReadOnlyBufferException (p. 2520) - If this buffer is read-only

6.314.3.21 `virtual FloatBuffer* decaf::nio::FloatBuffer::slice () const [pure virtual]`

Creates a new **FloatBuffer** (p. 1562) whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the newly create **FloatBuffer** (p. 1562) which the caller owns.

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1561).

6.314.3.22 `virtual std::string decaf::nio::FloatBuffer::toString () const [virtual]`

Returns

a std::string describing this object

6.314.3.23 `static FloatBuffer* decaf::nio::FloatBuffer::wrap (float * array, std::size_t offset, std::size_t length) throw (lang::exceptions::NullPointerException) [static]`

Wraps the passed buffer with a new **FloatBuffer** (p. 1562).

The new buffer will be backed by the given float array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be array.length, its position will be offset, its limit will be offset + length, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

array - The array that will back the new buffer

offset - The offset of the subarray to be used

length - The length of the subarray to be used

Returns

a new **FloatBuffer** (p. 1562) that is backed by buffer, caller owns.

6.314.3.24 `static FloatBuffer* decaf::nio::FloatBuffer::wrap (std::vector< float > & buffer) [static]`

Wraps the passed STL float Vector in a **FloatBuffer** (p. 1562).

The new buffer will be backed by the given float array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be buffer.size(), its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

buffer - The vector that will back the new buffer, the vector must have been sized to the desired size already by calling vector.resize(N).

Returns

a new **FloatBuffer** (p. 1562) that is backed by buffer, caller owns.

The documentation for this class was generated from the following file:

- src/main/decaf/nio/**FloatBuffer.h**

6.315 activemq::commands::FlushCommand Class Reference

```
#include <src/main/activemq/commands/FlushCommand.h>
```

Inheritance diagram for activemq::commands::FlushCommand:

Public Member Functions

- **FlushCommand** ()
- virtual **~FlushCommand** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **FlushCommand * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)

Copy the contents of the passed object into this object's members, overwriting any existing data.

- virtual std::string **toString** () const

*Returns a string containing the information for this **DataSet** (p. 1372) such as its type and value of its elements.*

- virtual bool **equals** (const **DataSet** *value) const

*Compares the **DataSet** (p. 1372) passed in to this one, and returns if they are equivalent.*

- virtual **Pointer**< **Command** > **visit** (**activemq::state::CommandVisitor** *visitor) throw (exceptions::ActiveMQException)

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_FLUSHCOMMAND** = 15

Protected Member Functions

- **FlushCommand** (const **FlushCommand** &)
- **FlushCommand** & **operator=** (const **FlushCommand** &)

6.315.1 Constructor & Destructor Documentation

6.315.1.1 **activemq::commands::FlushCommand::FlushCommand** (const **FlushCommand** &) [inline, protected]

6.315.1.2 **activemq::commands::FlushCommand::FlushCommand** ()

6.315.1.3 virtual **activemq::commands::FlushCommand::~~FlushCommand** () [virtual]

6.315.2 Member Function Documentation

6.315.2.1 virtual **FlushCommand*** **activemq::commands::FlushCommand::cloneDataSet** () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements **activemq::commands::DataSet** (p. 1373).

6.315.2.2 virtual void activemq::commands::FlushCommand::copyDataStructure (const DataStructure * *src*) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 598).

6.315.2.3 virtual bool activemq::commands::FlushCommand::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 598).

6.315.2.4 virtual unsigned char activemq::commands::FlushCommand::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataStructure** (p. 1372) type copy.

Implements **activemq::commands::DataStructure** (p. 1375).

6.315.2.5 FlushCommand& activemq::commands::FlushCommand::operator= (const FlushCommand &) [inline, protected]**6.315.2.6 virtual std::string activemq::commands::FlushCommand::toString () const [virtual]**

Returns a string containing the information for this **DataStructure** (p. 1372) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 602).

6.315.2.7 `virtual Pointer<Command> activemq::commands::FlushCommand::visit (activemq::state::CommandVisitor * visitor) throw (exceptions::ActiveMQException)` [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

Returns

a **Response** (p. 2611) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 995).

6.315.3 Field Documentation

6.315.3.1 `const unsigned char activemq::commands::FlushCommand::ID - FLUSHCOMMAND = 15` [static]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/FlushCommand.h`

6.316 activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 1576).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/FlushCommandMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller`:

Public Member Functions

- **FlushCommandMarshaller** ()
- virtual **~FlushCommandMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.316.1 Detailed Description

Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p.1576). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.316.2 Constructor & Destructor Documentation

6.316.2.1 activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller::FlushCommandMarshaller () [inline]

6.316.2.2 virtual activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller::~~FlushCommandMarshaller () [inline, virtual]

6.316.3 Member Function Documentation

6.316.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1331).

6.316.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller::getDataStructureType() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.316.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 631).

6.316.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 632).

6.316.3.5 virtual int activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 633).

6.316.3.6 virtual void activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 634).

6.316.3.7 virtual void activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 635).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/FlushCommandMarshaller.h`

6.317 `activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller` Class Reference

Marshaling code for Open Wire Format for `FlushCommandMarshaller` (p. 1579).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/FlushCommandMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller`:

Public Member Functions

- `FlushCommandMarshaller ()`
- `virtual ~FlushCommandMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaler.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshall an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.317.1 Detailed Description

Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p.1579). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.317.2 Constructor & Destructor Documentation

6.317.2.1 **activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller::FlushCommandMarshaller** () [inline]

6.317.2.2 **virtual**
activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller::~~FlushCommandMarshaller () [inline, virtual]

6.317.3 Member Function Documentation

6.317.3.1 **virtual commands::DataStructure*** **activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller::createObject** () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1331).

6.317.3.2 **virtual unsigned char** **activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller::getDataStructureType** () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.317.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 611).

6.317.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 612).

6.317.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 613).

```
6.317.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 614).

```
6.317.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 615).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/FlushCommandMarshaller.h`

6.318 activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 1583).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/FlushCommandMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller`:

Public Member Functions

- **FlushCommandMarshaller** ()
- virtual **~FlushCommandMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

- virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.318.1 Detailed Description

Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p.1583). NOTE! This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.318.2 Constructor & Destructor Documentation

6.318.2.1 activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller::FlushCommandMarshaller () [inline]

6.318.2.2 virtual activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller::~~FlushCommandMarshaller () [inline, virtual]

6.318.3 Member Function Documentation

6.318.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1331).

6.318.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1336).

6.318.3.3 virtual void activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 604).

6.318.3.4 virtual void activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 605).

6.318.3.5 virtual int activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 606).

```
6.318.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 608).

```
6.318.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 609).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/FlushCommandMarshaller.h`

6.319 activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 1587).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/FlushCommandMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller**:

Public Member Functions

- **FlushCommandMarshaller** ()
- virtual **~FlushCommandMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshall an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.319.1 Detailed Description

Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p.1587). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.319.2 Constructor & Destructor Documentation

6.319.2.1 activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller::FlushCommandMarshaller () [inline]

6.319.2.2 virtual activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller::~~FlushCommandMarshaller () [inline, virtual]

6.319.3 Member Function Documentation

6.319.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1331).

6.319.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1336).

6.319.3.3 virtual void activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 618).

6.319.3.4 virtual void activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 619).

6.319.3.5 virtual int activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 620).

```
6.319.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 621).

```
6.319.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 622).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/FlushCommandMarshaller.h`

6.320 activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 1591).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/FlushCommandMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller**:

Public Member Functions

- **FlushCommandMarshaller** ()
- virtual **~FlushCommandMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.320.1 Detailed Description

Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p.1591). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.320.2 Constructor & Destructor Documentation

6.320.2.1 activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller::FlushCommandMarshaller () [inline]

6.320.2.2 virtual
activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller::~~FlushCommandMarshaller () [inline, virtual]

6.320.3 Member Function Documentation

6.320.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1331).

6.320.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1336).

6.320.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 624).

6.320.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 625).

6.320.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 626).

```
6.320.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 628).

```
6.320.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 629).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/FlushCommandMarshaller.h`

6.321 decaf::util::logging::Formatter Class Reference

A **Formatter** (p. 1595) provides support for formatting LogRecords.

```
#include <src/main/decaf/util/logging/Formatter.h>
```

Inheritance diagram for `decaf::util::logging::Formatter`:

Public Member Functions

- virtual **~Formatter** ()
- virtual `std::string format (const LogRecord &record) const =0`
Format the given log record and return the formatted string.
- virtual `std::string formatMessage (const LogRecord &record) const =0`
Format the message string from a log record.
- virtual `std::string getHead (const Handler *handler)=0`
Return the header string for a set of formatted records.
- virtual `std::string getTail (const Handler *handler)=0`
Return the tail string for a set of formatted records.

6.321.1 Detailed Description

A **Formatter** (p. 1595) provides support for formatting LogRecords. Typically each logging **Handler** (p. 1604) will have a **Formatter** (p. 1595) associated with it. The **Formatter** (p. 1595) takes a **LogRecord** (p. 1928) and converts it to a string.

Some formatters (such as the XMLFormatter) need to wrap head and tail strings around a set of formatted records. The `getHeader` and `getTail` methods can be used to obtain these strings.

6.321.2 Constructor & Destructor Documentation

6.321.2.1 virtual decaf::util::logging::Formatter::~~Formatter () [inline, virtual]

6.321.3 Member Function Documentation

6.321.3.1 virtual std::string decaf::util::logging::Formatter::format (const LogRecord & *record*) const [pure virtual]

Format the given log record and return the formatted string.

Parameters

record The Log Record to Format

Returns

the formatted record.

Implemented in **decaf::util::logging::SimpleFormatter** (p.2783).

6.321.3.2 virtual std::string decaf::util::logging::Formatter::formatMessage (const LogRecord & *record*) const [pure virtual]

Format the message string from a log record.

Parameters

record The Log Record to Format

Returns

the formatted message

Implemented in **decaf::util::logging::SimpleFormatter** (p.2783).

6.321.3.3 virtual std::string decaf::util::logging::Formatter::getHead (const Handler * *handler*) [pure virtual]

Return the header string for a set of formatted records.

In the default implementation this method should return empty string

Parameters

handler the target handler, can be null

Returns

the head string

Implemented in **decaf::util::logging::SimpleFormatter** (p.2783).

6.321.3.4 virtual std::string decaf::util::logging::Formatter::getTail (const Handler * *handler*) [pure virtual]

Return the tail string for a set of formatted records.

In the default implementation this method should return empty string

Parameters

handler the target handler, can be null

Returns

the tail string

Implemented in **decaf::util::logging::SimpleFormatter** (p. 2783).

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/**Formatter.h**

6.322 decaf::util::concurrent::Future< V > Class Template Reference

A **Future** (p. 1597) represents the result of an asynchronous computation.

```
#include <src/main/decaf/util/concurrent/Future.h>
```

Public Member Functions

- virtual ~**Future** ()
- bool **cancel** (bool mayInterruptIfRunning)=0
Attempts to cancel execution of this task.
- bool **isCancelled** () const =0
Returns true if this task was canceled before it completed normally.
- bool **isDone** () const =0
Returns true if this task completed.
- V **get** ()=0 throw (CancellationException, InterruptedException, ExecutionException)
Waits if necessary for the computation to complete, and then retrieves its result.
- V **get** (long long timeout, **TimeUnit** unit)=0 throw (InterruptedException, ExecutionException, TimeoutException)
Waits if necessary for at most the given time for the computation to complete, and then retrieves its result, if available.

6.322.1 Detailed Description

`template<typename V> class decaf::util::concurrent::Future< V >`

A **Future** (p. 1597) represents the result of an asynchronous computation. Methods are provided to check if the computation is complete, to wait for its completion, and to retrieve the result of the computation. The result can only be retrieved using method `get` when the computation has completed, blocking if necessary until it is ready. Cancellation is performed by the `cancel` method. Additional methods are provided to determine if the task completed normally or was canceled. Once a computation has completed, the computation cannot be canceled. If you would like to use a **Future** (p. 1597) for the sake of cancellability but not provide a usable result, you can declare types of the form `Future<void*>` and return null as a result of the underlying task.

6.322.2 Constructor & Destructor Documentation

6.322.2.1 `template<typename V > virtual decaf::util::concurrent::Future< V >::~~Future () [inline, virtual]`

6.322.3 Member Function Documentation

6.322.3.1 `template<typename V > bool decaf::util::concurrent::Future< V >::cancel (bool mayInterruptIfRunning) [pure virtual]`

Attempts to cancel execution of this task.

This attempt will fail if the task has already completed, has already been canceled, or could not be canceled for some other reason. If successful, and this task has not started when `cancel` is called, this task should never run. If the task has already started, then the `mayInterruptIfRunning` parameter determines whether the thread executing this task should be interrupted in an attempt to stop the task.

After this method returns, subsequent calls to `isDone()` (p. 1600) will always return true. Subsequent calls to `isCancelled()` (p. 1600) will always return true if this method returned true.

Parameters

mayInterruptIfRunning - true if the thread executing this task should be interrupted; otherwise, in-progress tasks are allowed to complete.

Returns

false if the task could not be canceled, typically because it has already completed normally; true otherwise

6.322.3.2 `template<typename V > V decaf::util::concurrent::Future< V >::get (long long timeout, TimeUnit unit) throw (InterruptedException, ExecutionException, TimeoutException) [pure virtual]`

Waits if necessary for at most the given time for the computation to complete, and then retrieves its result, if available.

Parameters

timeout - the maximum time to wait

unit - the time unit of the timeout argument

Returns

the computed result

Exceptions

CancellationException (p. 898) - if the computation was canceled
ExecutionException (p. 1507) - if the computation threw an exception
InterruptedException - if the current thread was interrupted while waiting
TimeoutException (p. 2987) - if the wait timed out

6.322.3.3 `template<typename V > V decaf::util::concurrent::Future< V
 >::get () throw (CancellationException, InterruptedException,
 ExecutionException) [pure virtual]`

Waits if necessary for the computation to complete, and then retrieves its result.

Returns

the computed result.

Exceptions

CancellationException (p. 898) - if the computation was canceled
ExecutionException (p. 1507) - if the computation threw an exception
InterruptedException - if the current thread was interrupted while waiting

6.322.3.4 `template<typename V > bool decaf::util::concurrent::Future< V
 >::isCancelled () const [pure virtual]`

Returns true if this task was canceled before it completed normally.

Returns

true if this task was canceled before it completed

6.322.3.5 `template<typename V > bool decaf::util::concurrent::Future< V
 >::isDone () const [pure virtual]`

Returns true if this task completed.

Completion may be due to normal termination, an exception, or cancellation -- in all of these cases, this method will return true.

Returns

true if this task completed

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/Future.h`

6.323 activemq::transport::correlator::FutureResponse Class Reference

A container that holds a response object.

```
#include <src/main/activemq/transport/correlator/FutureResponse.h>
```

Public Member Functions

- **FutureResponse** ()
- virtual **~FutureResponse** ()
- virtual const **Pointer**< **Response** > & **getResponse** () const
Getters for the response property.
- virtual **Pointer**< **Response** > & **getResponse** ()
- virtual const **Pointer**< **Response** > & **getResponse** (unsigned int timeout) const
Getters for the response property.
- virtual **Pointer**< **Response** > & **getResponse** (unsigned int timeout)
- virtual void **setResponse** (const **Pointer**< **Response** > &response)
Setter for the response property.

6.323.1 Detailed Description

A container that holds a response object. Callers of the `getResponse` method will block until a response has been receive unless they call the `getRepsonse` that takes a timeout.

6.323.2 Constructor & Destructor Documentation

6.323.2.1 **activemq::transport::correlator::FutureResponse::FutureResponse** ()
[inline]

6.323.2.2 **virtual**
activemq::transport::correlator::FutureResponse::~~FutureResponse ()
[inline, virtual]

6.323.3 Member Function Documentation

6.323.3.1 **virtual const Pointer**<**Response**>& **activemq::transport::correlator::FutureResponse::getResponse** () const
[inline, virtual]

Getters for the response property.

Infinite Wait.

Returns

the response object for the request

- 6.323.3.2** `virtual Pointer<Response>& activemq::transport::correlator::FutureResponse::getResponse ()`
[inline, virtual]
- 6.323.3.3** `virtual Pointer<Response>& activemq::transport::correlator::FutureResponse::getResponse (unsigned int timeout)` [inline, virtual]
- 6.323.3.4** `virtual const Pointer<Response>& activemq::transport::correlator::FutureResponse::getResponse (unsigned int timeout) const` [inline, virtual]

Getters for the response property.

Timed Wait.

Parameters

timeout - time to wait in milliseconds

Returns

the response object for the request

- 6.323.3.5** `virtual void activemq::transport::correlator::FutureResponse::setResponse (const Pointer< Response > & response)` [inline, virtual]

Setter for the response property.

Parameters

response the response object for the request.

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/correlator/FutureResponse.h`

6.324 decaf::security::GeneralSecurityException Class Reference

```
#include <src/main/decaf/security/GeneralSecurityException.h>
```

Inheritance diagram for decaf::security::GeneralSecurityException:

Public Member Functions

- `GeneralSecurityException () throw ()`
Default Constructor.

- **GeneralSecurityException** (const Exception &ex) throw ()
Conversion Constructor from some other Exception.
- **GeneralSecurityException** (const **GeneralSecurityException** &ex) throw ()
Copy Constructor.
- **GeneralSecurityException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **GeneralSecurityException** (const std::exception *cause) throw ()
Constructor.
- **GeneralSecurityException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **GeneralSecurityException** * clone () const
Clones this exception.
- virtual ~**GeneralSecurityException** () throw ()

6.324.1 Constructor & Destructor Documentation

6.324.1.1 decaf::security::GeneralSecurityException::GeneralSecurityException () throw () [inline]

Default Constructor.

6.324.1.2 decaf::security::GeneralSecurityException::GeneralSecurityException (const Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

Parameters

ex An exception that should become this type of Exception

6.324.1.3 decaf::security::GeneralSecurityException::GeneralSecurityException (const GeneralSecurityException & ex) throw () [inline]

Copy Constructor.

Parameters

ex An exception that should become this type of Exception

6.324.1.4 `decaf::security::GeneralSecurityException::GeneralSecurityException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs
lineNumber The line number where the exception occurred.
cause The exception that was the cause for this one to be thrown.
msg The message to report
... list of primitives that are formatted into the message

6.324.1.5 `decaf::security::GeneralSecurityException::GeneralSecurityException (const std::exception * cause) throw () [inline]`

Constructor.

Parameters

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.324.1.6 `decaf::security::GeneralSecurityException::GeneralSecurityException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file name where exception occurs
lineNumber line number where the exception occurred.
msg message to report
... list of primitives that are formatted into the message

6.324.1.7 `virtual decaf::security::GeneralSecurityException::~GeneralSecurityException () throw () [inline, virtual]`

6.324.2 Member Function Documentation

6.324.2.1 `virtual GeneralSecurityException* decaf::security::GeneralSecurityException::clone () const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

A deep copy of this exception.

Reimplemented in **decaf::security::cert::CertificateEncodingException** (p. 906), **decaf::security::cert::CertificateException** (p. 908), **decaf::security::cert::CertificateExpiredException** (p. 910), **decaf::security::cert::CertificateNotYetValidException** (p. 912), **decaf::security::cert::CertificateParsingException** (p. 914), **decaf::security::InvalidKeyException** (p. 1700), **decaf::security::KeyException** (p. 1839), **decaf::security::NoSuchAlgorithmExceptionException** (p. 2274), **decaf::security::NoSuchProviderException** (p. 2279), and **decaf::security::SignatureException** (p. 2781).

The documentation for this class was generated from the following file:

- `src/main/decaf/security/GeneralSecurityException.h`

6.325 decaf::util::logging::Handler Class Reference

A **Handler** (p. 1604) object takes log messages from a **Logger** (p. 1908) and exports them.

`#include <src/main/decaf/util/logging/Handler.h>`

Inheritance diagram for `decaf::util::logging::Handler`:

Public Member Functions

- virtual `~Handler()`
- virtual void `flush()`=0
Flush the Handler's output, clears any buffers.
- virtual void `publish(const LogRecord &record)`=0
*Publish the Log Record to this **Handler** (p. 1604).*
- virtual void `isLoggable(const LogRecord &record)`=0
*Check if this **Handler** (p. 1604) would actually log a given **LogRecord** (p. 1928).*
- virtual void `setFilter(const Filter *filter)`=0
*Sets the **Filter** (p. 1527) that this **Handler** (p. 1604) uses to filter Log Records.*
- virtual const `Filter * getFilter()`=0
*Gets the **Filter** (p. 1527) that this **Handler** (p. 1604) uses to filter Log Records.*
- virtual void `setLevel(Level value)`=0
*Set (p. 2729) the log level specifying which message levels will be logged by this **Handler** (p. 1604).*

- virtual **Level** **getLevel** ()=0
*Get the log level specifying which message levels will be logged by this **Handler** (p. 1604).*
- virtual void **setFormatter** (const **Formatter** *formatter)=0
*Sets the **Formatter** (p. 1595) used by this **Handler** (p. 1604).*
- virtual const **Formatter** * **getFormatter** ()=0
*Gets the **Formatter** (p. 1595) used by this **Handler** (p. 1604).*

6.325.1 Detailed Description

A **Handler** (p.1604) object takes log messages from a **Logger** (p.1908) and exports them. It might for example, write them to a console or write them to a file, or send them to a network logging service, or forward them to an OS log, or whatever.

A **Handler** (p.1604) can be disabled by doing a **setLevel**(**Level.OFF**) and can be re-enabled by doing a **setLevel** with an appropriate level.

Handler (p. 1604) classes typically use **LogManager** (p.1923) properties to set default values for the Handler's **Filter** (p.1527), **Formatter** (p.1595), and **Level**. See the specific documentation for each concrete **Handler** (p.1604) class.

6.325.2 Constructor & Destructor Documentation

6.325.2.1 virtual **decaf::util::logging::Handler::~~Handler** () [inline, virtual]

6.325.3 Member Function Documentation

6.325.3.1 virtual void **decaf::util::logging::Handler::flush** () [pure virtual]

Flush the Handler's output, clears any buffers.

Implemented in **decaf::util::logging::StreamHandler** (p.2890).

6.325.3.2 virtual const **Filter*** **decaf::util::logging::Handler::getFilter** () [pure virtual]

Gets the **Filter** (p.1527) that this **Handler** (p.1604) uses to filter Log Records.

Returns

Filter (p.1527) derived instance

Implemented in **decaf::util::logging::StreamHandler** (p.2890).

6.325.3.3 virtual const **Formatter*** **decaf::util::logging::Handler::getFormatter** () [pure virtual]

Gets the **Formatter** (p.1595) used by this **Handler** (p.1604).

Returns

Filter (p. 1527) derived instance

Implemented in **decaf::util::logging::StreamHandler** (p. 2890).

6.325.3.4 virtual Level decaf::util::logging::Handler::getLevel () [pure virtual]

Get the log level specifying which message levels will be logged by this **Handler** (p. 1604).

Returns

Level enumeration value

Implemented in **decaf::util::logging::StreamHandler** (p. 2890).

6.325.3.5 virtual void decaf::util::logging::Handler::isLoggable (const LogRecord & record) [pure virtual]

Check if this **Handler** (p. 1604) would actually log a given **LogRecord** (p. 1928).

This method checks if the **LogRecord** (p. 1928) has an appropriate Level and whether it satisfies any **Filter** (p. 1527). It also may make other **Handler** (p. 1604) specific checks that might prevent a handler from logging the **LogRecord** (p. 1928).

Parameters

record LogRecord (p. 1928) to check

Implemented in **decaf::util::logging::StreamHandler** (p. 2891).

6.325.3.6 virtual void decaf::util::logging::Handler::publish (const LogRecord & record) [pure virtual]

Publish the Log Record to this **Handler** (p. 1604).

Parameters

record The Log Record to Publish

Implemented in **decaf::util::logging::StreamHandler** (p. 2891).

6.325.3.7 virtual void decaf::util::logging::Handler::setFilter (const Filter * filter) [pure virtual]

Sets the **Filter** (p. 1527) that this **Handler** (p. 1604) uses to filter Log Records.

For each call of publish the **Handler** (p. 1604) will call this **Filter** (p. 1527) (if it is non-null) to check if the **LogRecord** (p. 1928) should be published or discarded.

Parameters

filter Filter (p. 1527) derived instance

Implemented in **decaf::util::logging::StreamHandler** (p. 2891).

6.325.3.8 virtual void decaf::util::logging::Handler::setFormatter (const Formatter * *formatter*) [pure virtual]

Sets the **Formatter** (p. 1595) used by this **Handler** (p. 1604).

Some Handlers may not use Formatters, in which case the **Formatter** (p. 1595) will be remembered, but not used.

Parameters

formatter Filter (p. 1527) derived instance

Implemented in **decaf::util::logging::StreamHandler** (p. 2891).

6.325.3.9 virtual void decaf::util::logging::Handler::setLevel (Level *value*) [pure virtual]

Set (p. 2729) the log level specifying which message levels will be logged by this **Handler** (p. 1604).

The intention is to allow developers to turn on voluminous logging, but to limit the messages that are sent to certain Handlers.

Parameters

value Level enumeration value

Implemented in **decaf::util::logging::StreamHandler** (p. 2892).

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/**Handler.h**

6.326 decaf::internal::util::HexStringParser Class Reference

```
#include <src/main/decaf/internal/util/HexStringParser.h>
```

Public Member Functions

- **HexStringParser** (int exponentWidth, int mantissaWidth)
Create a new HexParser.
- virtual ~**HexStringParser** ()
- long long **parse** (const std::string &hexString)
Parses a hex string using the specs given in the constructor and returns a long long with the bits of the parsed string, the caller can then convert those to a float or double as needed.

Static Public Member Functions

- static double **parseDouble** (const std::string &hexString)
- static float **parseFloat** (const std::string &hexString)

6.326.1 Constructor & Destructor Documentation

6.326.1.1 decaf::internal::util::HexStringParser::HexStringParser (int *exponentWidth*, int *mantissaWidth*)

Create a new HexParser.

Parameters

exponentWidth - Width of the exponent for the type to parse

mantissaWidth - Width of the mantissa for the type to parse

6.326.1.2 virtual decaf::internal::util::HexStringParser::~~HexStringParser () [inline, virtual]

6.326.2 Member Function Documentation

6.326.2.1 long long decaf::internal::util::HexStringParser::parse (const std::string & *hexString*)

Parses a hex string using the specs given in the constructor and returns a long long with the bits of the parsed string, the caller can then convert those to a float or double as needed.

Parameters

hexString - string to parse

Returns

the bits parsed from the string

6.326.2.2 static double decaf::internal::util::HexStringParser::parseDouble (const std::string & *hexString*) [static]

6.326.2.3 static float decaf::internal::util::HexStringParser::parseFloat (const std::string & *hexString*) [static]

The documentation for this class was generated from the following file:

- src/main/decaf/internal/util/HexStringParser.h

6.327 activemq::wireformat::openwire::utils::HexTable Class Reference

The **HexTable** (p. 1609) class maps hexadecimal strings to the value of an index into the table, i.e.

```
#include <src/main/activemq/wireformat/openwire/utils/HexTable.h>
```

Public Member Functions

- **HexTable** ()
- virtual **~HexTable** ()
- virtual const std::string & **operator[]** (std::size_t index) throw (decaf::lang::exceptions::IndexOutOfBoundsException)

Index operator for this Table, will throw an exception if the index requested is out of bounds for this table.

- virtual const std::string & **operator[]** (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException)
- virtual std::size_t **size** () const

Returns the max size of this Table.

6.327.1 Detailed Description

The **HexTable** (p. 1609) class maps hexadecimal strings to the value of an index into the table, i.e. the class will return "FF" for the index 255 in the table.

6.327.2 Constructor & Destructor Documentation

6.327.2.1 activemq::wireformat::openwire::utils::HexTable::HexTable ()

6.327.2.2 virtual activemq::wireformat::openwire::utils::HexTable::~~HexTable ()
[inline, virtual]

6.327.3 Member Function Documentation

6.327.3.1 virtual const std::string&
activemq::wireformat::openwire::utils::HexTable::operator[] (std::size_t
index) throw (decaf::lang::exceptions::IndexOutOfBoundsException)
[virtual]

Index operator for this Table, will throw an exception if the index requested is out of bounds for this table.

Parameters

index The index of the value in the table to fetch.

Returns

string containing the hex value if the index

Exceptions

IndexOutOfBoundsException

6.327.3.2 `virtual const std::string& activemq::wireformat::openwire::utils::HexTable::operator[] (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException) [virtual]`

6.327.3.3 `virtual std::size_t activemq::wireformat::openwire::utils::HexTable::size () const [inline, virtual]`

Returns the max size of this Table.

Returns

an integer size value

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/utils/HexTable.h`

6.328 decaf::net::HttpRetryException Class Reference

```
#include <src/main/decaf/net/HttpRetryException.h>
```

Inheritance diagram for decaf::net::HttpRetryException:

Public Member Functions

- **HttpRetryException** () throw ()
Default Constructor.
- **HttpRetryException** (const Exception &ex) throw ()
Conversion Constructor from some other Exception.
- **HttpRetryException** (const **HttpRetryException** &ex) throw ()
Copy Constructor.
- **HttpRetryException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **HttpRetryException** (const std::exception *cause) throw ()
Constructor.
- **HttpRetryException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **HttpRetryException** * **clone** () const
Clones this exception.
- virtual ~**HttpRetryException** () throw ()

6.328.1 Constructor & Destructor Documentation

6.328.1.1 `decaf::net::HttpRetryException::HttpRetryException () throw ()` [inline]

Default Constructor.

6.328.1.2 `decaf::net::HttpRetryException::HttpRetryException (const Exception & ex) throw ()` [inline]

Conversion Constructor from some other Exception.

Parameters

ex An exception that should become this type of Exception

References `decaf::lang::Exception::Exception()`.

6.328.1.3 `decaf::net::HttpRetryException::HttpRetryException (const HttpRetryException & ex) throw ()` [inline]

Copy Constructor.

Parameters

ex An exception that should become this type of Exception

References `decaf::lang::Exception::Exception()`.

6.328.1.4 `decaf::net::HttpRetryException::HttpRetryException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw ()` [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.328.1.5 `decaf::net::HttpRetryException::HttpRetryException (const std::exception * cause) throw ()` [inline]

Constructor.

Parameters

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.328.1.6 `decaf::net::HttpRetryException::HttpRetryException (const char * file,
const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.328.1.7 `virtual decaf::net::HttpRetryException::~~HttpRetryException ()
throw () [inline, virtual]`

6.328.2 Member Function Documentation

6.328.2.1 `virtual HttpRetryException* decaf::net::HttpRetryException::clone ()
const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

a new Exception instance that is a copy of this Exception object.

Reimplemented from `decaf::io::IOException` (p. 1709).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/HttpRetryException.h`

6.329 decaf::lang::exceptions::IllegalArgumentException Class Reference

```
#include <src/main/decaf/lang/exceptions/IllegalArgumentException.h>
```

Inheritance diagram for `decaf::lang::exceptions::IllegalArgumentException`:

Public Member Functions

- `IllegalArgumentException () throw ()`
Default Constructor.

- **IllegalArgumentException** (const **Exception** &ex) throw ()
*Conversion Constructor from some other **Exception** (p. 1476).*
- **IllegalArgumentException** (const **IllegalArgumentException** &ex) throw ()
Copy Constructor.
- **IllegalArgumentException** (const std::exception *cause) throw ()
Constructor.
- **IllegalArgumentException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **IllegalArgumentException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **IllegalArgumentException** * clone () const
Clones this exception.
- virtual ~**IllegalArgumentException** () throw ()

6.329.1 Constructor & Destructor Documentation

6.329.1.1 decaf::lang::exceptions::IllegalArgumentException::IllegalArgumentException () throw () [inline]

Default Constructor.

6.329.1.2 decaf::lang::exceptions::IllegalArgumentException::IllegalArgumentException (const Exception & ex) throw () [inline]

Conversion Constructor from some other **Exception** (p. 1476).

Parameters

ex The **Exception** (p. 1476) whose data is to be copied into this one.

6.329.1.3 decaf::lang::exceptions::IllegalArgumentException::IllegalArgumentException (const IllegalArgumentException & ex) throw () [inline]

Copy Constructor.

Parameters

ex The **Exception** (p. 1476) whose data is to be copied into this one.

6.329.1.4 decaf::lang::exceptions::IllegalArgumentException::IllegalArgumentException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters

***cause* Pointer** (p.2343) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.329.1.5 decaf::lang::exceptions::IllegalArgumentException::IllegalArgumentException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.329.1.6 decaf::lang::exceptions::IllegalArgumentException::IllegalArgumentException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.329.1.7 virtual decaf::lang::exceptions::IllegalArgumentException::~IllegalArgumentException () throw () [inline, virtual]

6.329.2 Member Function Documentation

6.329.2.1 virtual IllegalArgumentException* decaf::lang::exceptions::IllegalArgumentException::clone () const [inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

an new **Exception** (p. 1476) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1479).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/exceptions/IllegalArgumentException.h`

6.330 decaf::lang::exceptions::IllegalMonitorStateException Class Reference

```
#include <src/main/decaf/lang/exceptions/IllegalMonitorStateException.h>
```

Inheritance diagram for `decaf::lang::exceptions::IllegalMonitorStateException`:

Public Member Functions

- **IllegalMonitorStateException** () throw ()
Default Constructor.
- **IllegalMonitorStateException** (const **Exception** &ex) throw ()
*Conversion Constructor from some other **Exception** (p. 1476).*
- **IllegalMonitorStateException** (const **IllegalMonitorStateException** &ex) throw ()
Copy Constructor.
- **IllegalMonitorStateException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **IllegalMonitorStateException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **IllegalMonitorStateException** (const std::exception *cause) throw ()
Constructor.
- virtual **IllegalMonitorStateException** * clone () const
Clones this exception.
- virtual ~**IllegalMonitorStateException** () throw ()

6.330.1 Constructor & Destructor Documentation

6.330.1.1 decaf::lang::exceptions::IllegalMonitorStateException::IllegalMonitorStateException
() throw () [inline]

Default Constructor.

6.330.1.2 decaf::lang::exceptions::IllegalMonitorStateException::IllegalMonitorStateException
(const Exception & *ex*) throw () [inline]

Conversion Constructor from some other **Exception** (p. 1476).

Parameters

ex The **Exception** (p. 1476) whose data is to be copied into this one.

6.330.1.3 decaf::lang::exceptions::IllegalMonitorStateException::IllegalMonitorStateException
(const IllegalMonitorStateException & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex The **Exception** (p. 1476) whose data is to be copied into this one.

6.330.1.4 decaf::lang::exceptions::IllegalMonitorStateException::IllegalMonitorStateException
(const char * *file*, const int *lineNumber*, const char * *msg*, ...)
throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.330.1.5 decaf::lang::exceptions::IllegalMonitorStateException::IllegalMonitorStateException
(const char * *file*, const int *lineNumber*, const std::exception *
cause, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.330.1.6 `decaf::lang::exceptions::IllegalMonitorStateException::IllegalMonitorStateException (const std::exception * cause) throw () [inline]`

Constructor.

Parameters

cause **Pointer** (p. 2343) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.330.1.7 `virtual decaf::lang::exceptions::IllegalMonitorStateException::~~IllegalMonitorStateException () throw () [inline, virtual]`

6.330.2 Member Function Documentation

6.330.2.1 `virtual IllegalMonitorStateException* decaf::lang::exceptions::IllegalMonitorStateException::clone () const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

an new **Exception** (p. 1476) instance that is a copy of this one.

Reimplemented from `decaf::lang::Exception` (p. 1479).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/exceptions/IllegalMonitorStateException.h`

6.331 decaf::lang::exceptions::IllegalStateException Class Reference

```
#include <src/main/decaf/lang/exceptions/IllegalStateException.h>
```

Inheritance diagram for `decaf::lang::exceptions::IllegalStateException`:

Public Member Functions

- **IllegalStateException** () throw ()
Default Constructor.
- **IllegalStateException** (const **Exception** &ex) throw ()
*Conversion Constructor from some other **Exception** (p. 1476).*
- **IllegalStateException** (const **IllegalStateException** &ex) throw ()
Copy Constructor.
- **IllegalStateException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **IllegalStateException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **IllegalStateException** (const std::exception *cause) throw ()
Constructor.
- virtual **IllegalStateException** * clone () const
Clones this exception.
- virtual ~**IllegalStateException** () throw ()

6.331.1 Constructor & Destructor Documentation

6.331.1.1 decaf::lang::exceptions::IllegalStateException::IllegalStateException ()
throw () [inline]

Default Constructor.

6.331.1.2 decaf::lang::exceptions::IllegalStateException::IllegalStateException (const **Exception** & ex) throw () [inline]

Conversion Constructor from some other **Exception** (p. 1476).

Parameters

ex The **Exception** (p. 1476) whose data is to be copied into this one.

6.331.1.3 decaf::lang::exceptions::IllegalStateException::IllegalStateException (const **IllegalStateException** & ex) throw () [inline]

Copy Constructor.

Parameters

ex The **Exception** (p. 1476) whose data is to be copied into this one.

6.331.1.4 `decaf::lang::exceptions::IllegalStateException::IllegalStateException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.331.1.5 `decaf::lang::exceptions::IllegalStateException::IllegalStateException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.331.1.6 `decaf::lang::exceptions::IllegalStateException::IllegalStateException (const std::exception * cause) throw () [inline]`

Constructor.

Parameters

cause **Pointer** (p. 2343) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.331.1.7 `virtual decaf::lang::exceptions::IllegalStateException::~~IllegalStateException () throw () [inline, virtual]`

6.331.2 Member Function Documentation

6.331.2.1 `virtual IllegalStateException* decaf::lang::exceptions::IllegalStateException::clone () const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

an new **Exception** (p. 1476) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1479).

Reimplemented in **decaf::nio::InvalidMarkException** (p. 1703).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/exceptions/IllegalStateException.h`

6.332 cms::IllegalStateException Class Reference

This exception is thrown when a method is invoked at an illegal or inappropriate time or if the provider is not in an appropriate state for the requested operation.

```
#include <src/main/cms/IllegalStateException.h>
```

Inheritance diagram for cms::IllegalStateException:

Public Member Functions

- **IllegalStateException** () throw ()
- **IllegalStateException** (const **IllegalStateException** &ex) throw ()
- **IllegalStateException** (const std::string &message, const std::exception *cause) throw ()
- **IllegalStateException** (const std::string &message, const std::exception *cause, const std::vector< std::pair< std::string, int > > &stackTrace) throw ()
- virtual ~**IllegalStateException** () throw ()

6.332.1 Detailed Description

This exception is thrown when a method is invoked at an illegal or inappropriate time or if the provider is not in an appropriate state for the requested operation. For example, this exception must be thrown if **Session.commit** (p. 2667) is called on a non-transacted session.

Since

1.3

6.332.2 Constructor & Destructor Documentation

- 6.332.2.1 `cms::IllegalStateException::IllegalStateException () throw ()`
- 6.332.2.2 `cms::IllegalStateException::IllegalStateException (const
IllegalStateException & ex) throw ()`
- 6.332.2.3 `cms::IllegalStateException::IllegalStateException (const std::string &
message, const std::exception * cause) throw ()`
- 6.332.2.4 `cms::IllegalStateException::IllegalStateException (const std::string &
message, const std::exception * cause, const std::vector< std::pair<
std::string, int > > & stackTrace) throw ()`
- 6.332.2.5 `virtual cms::IllegalStateException::~~IllegalStateException () throw ()
[virtual]`

The documentation for this class was generated from the following file:

- `src/main/cms/IllegalStateException.h`

6.333 decaf::lang::exceptions::IllegalThreadStateException Class Reference

```
#include <src/main/decaf/lang/exceptions/IllegalThreadStateException.h>
```

Inheritance diagram for `decaf::lang::exceptions::IllegalThreadStateException`:

Public Member Functions

- `IllegalThreadStateException () throw ()`
Default Constructor.
- `IllegalThreadStateException (const Exception &ex) throw ()`
*Conversion Constructor from some other **Exception** (p. 1476).*
- `IllegalThreadStateException (const IllegalThreadStateException &ex) throw ()`
Copy Constructor.
- `IllegalThreadStateException (const char *file, const int lineNumber, const char *msg,...)
throw ()`
Constructor - Initializes the file name and line number where this message occurred.
- `IllegalThreadStateException (const char *file, const int lineNumber, const std::exception
*cause, const char *msg,...) throw ()`
Constructor - Initializes the file name and line number where this message occurred.
- `IllegalThreadStateException (const std::exception *cause) throw ()`

Constructor.

- virtual **IllegalThreadStateException** * **clone** () const
Clones this exception.
- virtual ~**IllegalThreadStateException** () throw ()

6.333.1 Constructor & Destructor Documentation

6.333.1.1 decaf::lang::exceptions::IllegalThreadStateException::IllegalThreadStateException () throw () [inline]

Default Constructor.

6.333.1.2 decaf::lang::exceptions::IllegalThreadStateException::IllegalThreadStateException (const Exception & *ex*) throw () [inline]

Conversion Constructor from some other **Exception** (p. 1476).

Parameters

ex The **Exception** (p. 1476) whose data is to be copied into this one.

6.333.1.3 decaf::lang::exceptions::IllegalThreadStateException::IllegalThreadStateException (const IllegalThreadStateException & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex The **Exception** (p. 1476) whose data is to be copied into this one.

6.333.1.4 decaf::lang::exceptions::IllegalThreadStateException::IllegalThreadStateException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.333.1.5 `decaf::lang::exceptions::IllegalThreadStateException::IllegalThreadStateException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.333.1.6 `decaf::lang::exceptions::IllegalThreadStateException::IllegalThreadStateException (const std::exception * cause) throw () [inline]`

Constructor.

Parameters

cause **Pointer** (p. 2343) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.333.1.7 `virtual decaf::lang::exceptions::IllegalThreadStateException::~~IllegalThreadStateException () throw () [inline, virtual]`

6.333.2 Member Function Documentation

6.333.2.1 `virtual IllegalThreadStateException* decaf::lang::exceptions::IllegalThreadStateException::clone () const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

an new **Exception** (p. 1476) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1479).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/exceptions/IllegalThreadStateException.h`

6.334 activemq::transport::inactivity::InactivityMonitor Class Reference

```
#include <src/main/activemq/transport/inactivity/InactivityMonitor.h>
```

Inheritance diagram for activemq::transport::inactivity::InactivityMonitor:

Public Member Functions

- **InactivityMonitor** (const **Pointer**< **Transport** > &next, const **Pointer**< **wireformat::WireFormat** > &wireFormat)
- **InactivityMonitor** (const **Pointer**< **Transport** > &next, const **decaf::util::Properties** &properties, const **Pointer**< **wireformat::WireFormat** > &wireFormat)
- virtual **~InactivityMonitor** ()
- virtual void **close** () throw (**decaf::io::IOException**)
Stops the polling thread and closes the streams.
- virtual void **onException** (const **decaf::lang::Exception** &ex)
Event handler for an exception from a command transport.
- virtual void **onCommand** (const **Pointer**< **Command** > &command)
Event handler for the receipt of a command.
- virtual void **oneway** (const **Pointer**< **Command** > &command) throw (**decaf::io::IOException**, **decaf::lang::exceptions::UnsupportedOperationException**)
Sends a one-way command.
- bool **isKeepAliveResponseRequired** () const
- void **setKeepAliveResponseRequired** (bool value)
- long long **getReadCheckTime** () const
- void **setReadCheckTime** (long long value)
- long long **getWriteCheckTime** () const
- void **setWriteCheckTime** (long long value)
- long long **getInitialDelayTime** () const
- void **setInitialDelayTime** (long long value) const

Friends

- class **ReadChecker**
- class **AsyncSignalReadErrorTask**
- class **WriteChecker**
- class **AsyncWriteTask**

6.334.1 Constructor & Destructor Documentation

- 6.334.1.1** `activemq::transport::inactivity::InactivityMonitor::InactivityMonitor (const Pointer< Transport > & next, const Pointer< wireformat::WireFormat > & wireFormat)`
- 6.334.1.2** `activemq::transport::inactivity::InactivityMonitor::InactivityMonitor (const Pointer< Transport > & next, const decaf::util::Properties & properties, const Pointer< wireformat::WireFormat > & wireFormat)`
- 6.334.1.3** `virtual
activemq::transport::inactivity::InactivityMonitor::~~InactivityMonitor () [virtual]`

6.334.2 Member Function Documentation

- 6.334.2.1** `virtual void activemq::transport::inactivity::InactivityMonitor::close ()
throw (decaf::io::IOException) [virtual]`

Stops the polling thread and closes the streams.

This can be called explicitly, but is also called in the destructor. Once this object has been closed, it cannot be restarted.

Exceptions

IOException if an error occurs while closing the **Transport** (p. 3066).

Reimplemented from **activemq::transport::TransportFilter** (p. 3075).

- 6.334.2.2** `long long ac-
tivemq::transport::inactivity::InactivityMonitor::getInitialDelayTime () const`
- 6.334.2.3** `long long ac-
tivemq::transport::inactivity::InactivityMonitor::getReadCheckTime () const`
- 6.334.2.4** `long long ac-
tivemq::transport::inactivity::InactivityMonitor::getWriteCheckTime () const`
- 6.334.2.5** `bool ac-
tivemq::transport::inactivity::InactivityMonitor::isKeepAliveResponseRequired () const`
- 6.334.2.6** `virtual void ac-
tivemq::transport::inactivity::InactivityMonitor::onCommand (const
Pointer< Command > & command) [virtual]`

Event handler for the receipt of a command.

Parameters

command - the received command object.

Reimplemented from `activemq::transport::TransportFilter` (p. 3077).

6.334.2.7 `virtual void activemq::transport::inactivity::InactivityMonitor::oneway (const Pointer< Command > & command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Sends a one-way command.

Does not wait for any response from the broker.

Parameters

command the command to be sent.

Exceptions

IOException if an exception occurs during writing of the command.

UnsupportedOperationException if this method is not implemented by this transport.

Reimplemented from `activemq::transport::TransportFilter` (p. 3077).

6.334.2.8 `virtual void activemq::transport::inactivity::InactivityMonitor::onException (const decaf::lang::Exception & ex) [virtual]`

Event handler for an exception from a command transport.

Parameters

ex The exception to handle.

Reimplemented from `activemq::transport::TransportFilter` (p. 3078).

- 6.334.2.9 `void activemq::transport::inactivity::InactivityMonitor::setInitialDelayTime (long long value) const`
- 6.334.2.10 `void activemq::transport::inactivity::InactivityMonitor::setKeepAliveResponseRequired (bool value)`
- 6.334.2.11 `void activemq::transport::inactivity::InactivityMonitor::setReadCheckTime (long long value)`
- 6.334.2.12 `void activemq::transport::inactivity::InactivityMonitor::setWriteCheckTime (long long value)`

6.334.3 Friends And Related Function Documentation

- 6.334.3.1 `friend class AsyncSignalReadErrorTask [friend]`
- 6.334.3.2 `friend class AsyncWriteTask [friend]`
- 6.334.3.3 `friend class ReadChecker [friend]`
- 6.334.3.4 `friend class WriteChecker [friend]`

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/inactivity/InactivityMonitor.h`

6.335 decaf::lang::exceptions::IndexOutOfBoundsException Class Reference

```
#include <src/main/decaf/lang/exceptions/IndexOutOfBoundsException.h>
```

Inheritance diagram for `decaf::lang::exceptions::IndexOutOfBoundsException`:

Public Member Functions

- `IndexOutOfBoundsException () throw ()`
Default Constructor.
- `IndexOutOfBoundsException (const Exception &ex) throw ()`
*Conversion Constructor from some other **Exception** (p. 1476).*
- `IndexOutOfBoundsException (const IndexOutOfBoundsException &ex) throw ()`
Copy Constructor.

- **IndexOutOfBoundsException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **IndexOutOfBoundsException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **IndexOutOfBoundsException** (const std::exception *cause) throw ()
Constructor.
- virtual **IndexOutOfBoundsException** * clone () const
Clones this exception.
- virtual ~**IndexOutOfBoundsException** () throw ()

6.335.1 Constructor & Destructor Documentation

6.335.1.1 decaf::lang::exceptions::IndexOutOfBoundsException::IndexOutOfBoundsException () throw () [inline]

Default Constructor.

6.335.1.2 decaf::lang::exceptions::IndexOutOfBoundsException::IndexOutOfBoundsException (const Exception & ex) throw () [inline]

Conversion Constructor from some other **Exception** (p. 1476).

Parameters

ex The **Exception** (p. 1476) whose data is to be copied into this one.

6.335.1.3 decaf::lang::exceptions::IndexOutOfBoundsException::IndexOutOfBoundsException (const IndexOutOfBoundsException & ex) throw () [inline]

Copy Constructor.

Parameters

ex The **Exception** (p. 1476) whose data is to be copied into this one.

6.335.1.4 decaf::lang::exceptions::IndexOutOfBoundsException::IndexOutOfBoundsException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs
lineNumber The line number where the exception occurred.
msg The message to report
... list of primitives that are formatted into the message

6.335.1.5 `decaf::lang::exceptions::IndexOutOfBoundsException::IndexOutOfBoundsException`
(`const char * file`, `const int lineNumber`, `const std::exception * cause`, `const char * msg`, ...) `throw ()` [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs
lineNumber The line number where the exception occurred.
cause The exception that was the cause for this one to be thrown.
msg The message to report
... list of primitives that are formatted into the message

6.335.1.6 `decaf::lang::exceptions::IndexOutOfBoundsException::IndexOutOfBoundsException`
(`const std::exception * cause`) `throw ()` [inline]

Constructor.

Parameters

cause **Pointer** (p. 2343) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.335.1.7 `virtual`
`decaf::lang::exceptions::IndexOutOfBoundsException::~~IndexOutOfBoundsException`
() `throw ()` [inline, virtual]

6.335.2 Member Function Documentation

6.335.2.1 `virtual IndexOutOfBoundsException* decaf::lang::exceptions::IndexOutOfBoundsException::clone` () `const` [inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

an new **Exception** (p. 1476) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1479).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/exceptions/IndexOutOfBoundsException.h`

6.336 decaf::io::InputStream Class Reference

Base interface for an input stream.

```
#include <src/main/decaf/io/InputStream.h>
```

Inheritance diagram for decaf::io::InputStream:

Public Member Functions

- virtual **~InputStream** ()
- virtual void **mark** (int readLimit)=0
Marks the current position in the stream. A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.
- virtual void **reset** ()=0 throw (IOException)
Repositions this stream to the position at the time the mark method was last called on this input stream.
- virtual bool **markSupported** () const =0
Determines if this input stream supports the mark and reset methods.
- virtual std::size_t **available** () const =0 throw (IOException)
Indicates the number of bytes available.
- virtual int **read** ()=0 throw (IOException)
Reads a single byte from the buffer.
- virtual int **read** (unsigned char *buffer, std::size_t offset, std::size_t bufferSize)=0 throw (IOException, lang::exceptions::NullPointerException)
Reads an array of bytes from the buffer.
- virtual std::size_t **skip** (std::size_t num)=0 throw (io::IOException, lang::exceptions::UnsupportedOperationException)
Skips over and discards n bytes of data from this input stream.

6.336.1 Detailed Description

Base interface for an input stream.

6.336.2 Constructor & Destructor Documentation

6.336.2.1 `virtual decaf::io::InputStream::~~InputStream () [inline, virtual]`

6.336.3 Member Function Documentation

6.336.3.1 `virtual std::size_t decaf::io::InputStream::available () const throw (IOException) [pure virtual]`

Indicates the number of bytes available.

Returns

the number of bytes available on this input stream.

Exceptions

IOException (p. 1707) if an error occurs.

Implemented in `decaf::internal::io::StandardInputStream` (p. 2820), `decaf::io::BlockingByteArrayInputStream` (p. 668), `decaf::io::BufferedInputStream` (p. 749), `decaf::io::ByteArrayInputStream` (p. 830), `decaf::io::FilterInputStream` (p. 1530), and `decaf::net::SocketInputStream` (p. 2798).

6.336.3.2 `virtual void decaf::io::InputStream::mark (int readLimit) [pure virtual]`

Marks the current position in the stream. A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.

If a stream instance reports that marks are supported then the stream will ensure that the same bytes can be read again after the reset method is called so long the readLimit is not reached.

Parameters

readLimit - max bytes read before marked position is invalid.

Implemented in `decaf::internal::io::StandardInputStream` (p. 2820), `decaf::io::BlockingByteArrayInputStream` (p. 669), `decaf::io::BufferedInputStream` (p. 750), `decaf::io::ByteArrayInputStream` (p. 831), `decaf::io::FilterInputStream` (p. 1531), and `decaf::net::SocketInputStream` (p. 2799).

6.336.3.3 `virtual bool decaf::io::InputStream::markSupported () const [pure virtual]`

Determines if this input stream supports the mark and reset methods.

Whether or not mark and reset are supported is an invariant property of a particular input stream instance.

Returns

true if this stream instance supports marks

Implemented in `decaf::internal::io::StandardInputStream` (p. 2821), `decaf::io::BlockingByteArrayInputStream` (p. 669), `decaf::io::BufferedInputStream` (p. 750), `decaf::io::ByteArrayInputStream` (p. 831), `decaf::io::FilterInputStream` (p. 1531), and `decaf::net::SocketInputStream` (p. 2799).

6.336.3.4 `virtual int decaf::io::InputStream::read () throw (IOException)` [pure virtual]

Reads a single byte from the buffer.

The value byte is returned as an int in the range 0 to 255. If no byte is available because the end of the stream has been reached, the value -1 is returned. This method blocks until input data is available, the end of the stream is detected, or an exception is thrown.

Returns

The next byte or -1 if the end of stream is reached.

Exceptions

IOException (p. 1707) thrown if an error occurs.

Implemented in `activemq::io::LoggingInputStream` (p. 1918), `decaf::internal::io::StandardInputStream` (p. 2821), `decaf::io::BlockingByteArrayInputStream` (p. 670), `decaf::io::BufferedInputStream` (p. 751), `decaf::io::ByteArrayInputStream` (p. 832), `decaf::io::DataInputStream` (p. 1293), `decaf::io::FilterInputStream` (p. 1532), and `decaf::net::SocketInputStream` (p. 2800).

6.336.3.5 `virtual int decaf::io::InputStream::read (unsigned char * buffer, std::size_t offset, std::size_t bufferSize) throw (IOException, lang::exceptions::NullPointerException)` [pure virtual]

Reads an array of bytes from the buffer.

Blocks until the requested number of bytes are available.

Parameters

buffer (out) the target buffer.

offset the position in the buffer to start reading from.

bufferSize the size of the output buffer.

Returns

The number of bytes read or -1 if EOF is detected

Exceptions

IOException (p. 1707) thrown if an error occurs.

NullPointerException if buffer is null

Implemented in `activemq::io::LoggingInputStream` (p. 1918), `decaf::internal::io::StandardInputStream` (p. 2822), `decaf::io::BlockingByteArrayInputStream` (p. 670), `decaf::io::BufferedInputStream` (p. 750), `decaf::io::ByteArrayInputStream` (p. 832), `decaf::io::DataInputStream` (p. 1294), `decaf::io::FilterInputStream` (p. 1533), and `decaf::net::SocketInputStream` (p. 2800).

6.336.3.6 `virtual void decaf::io::InputStream::reset () throw (IOException)` [pure virtual]

Repositions this stream to the position at the time the mark method was last called on this input stream.

If the method `markSupported` returns true, then: * If the method `mark` has not been called since the stream was created, or the number of bytes read from the stream since `mark` was last called is larger than the argument to `mark` at that last call, then an **IOException** (p. 1707) might be thrown. * If such an **IOException** (p. 1707) is not thrown, then the stream is reset to a state such that all the bytes read since the most recent call to `mark` (or since the start of the file, if `mark` has not been called) will be resupplied to subsequent callers of the `read` method, followed by any bytes that otherwise would have been the next input data as of the time of the call to `reset`. If the method `markSupported` returns false, then: * The call to `reset` may throw an **IOException** (p. 1707). * If an **IOException** (p. 1707) is not thrown, then the stream is reset to a fixed state that depends on the particular type of the input stream and how it was created. The bytes that will be supplied to subsequent callers of the `read` method depend on the particular type of the input stream.

Exceptions

IOException (p. 1707)

Implemented in `decaf::internal::io::StandardInputStream` (p. 2822), `decaf::io::BlockingByteArrayInputStream` (p. 671), `decaf::io::BufferedInputStream` (p. 751), `decaf::io::ByteArrayInputStream` (p. 833), `decaf::io::FilterInputStream` (p. 1533), and `decaf::net::SocketInputStream` (p. 2801).

6.336.3.7 `virtual std::size_t decaf::io::InputStream::skip (std::size_t num) throw (io::IOException, lang::exceptions::UnsupportedOperationException)` [pure virtual]

Skips over and discards `n` bytes of data from this input stream.

The `skip` method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before `n` bytes have been skipped is only one possibility. The actual number of bytes skipped is returned. If `n` is negative, no bytes are skipped.

The `skip` method of **InputStream** (p. 1630) creates a byte array and then repeatedly reads into it until `n` bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.

Parameters

num - the number of bytes to skip

Returns

total bytes skipped

Exceptions

IOException (p. 1707) if an error occurs

Implemented in `decaf::internal::io::StandardInputStream` (p. 2823), `decaf::io::BlockingByteArrayInputStream` (p. 671), `decaf::io::BufferedInputStream`

(p. 751), `decaf::io::ByteArrayInputStream` (p. 833), `decaf::io::DataInputStream` (p. 1300), `decaf::io::FilterInputStream` (p. 1534), and `decaf::net::SocketInputStream` (p. 2801).

The documentation for this class was generated from the following file:

- `src/main/decaf/io/InputStream.h`

6.337 decaf::internal::nio::IntArrayBuffer Class Reference

```
#include <src/main/decaf/internal/nio/IntArrayBuffer.h>
```

Inheritance diagram for `decaf::internal::nio::IntArrayBuffer`:

Public Member Functions

- **IntArrayBuffer** (std::size_t capacity, bool readOnly=false)
*Creates a **IntArrayBuffer** (p. 1634) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*
- **IntArrayBuffer** (int *array, std::size_t offset, std::size_t capacity, bool readOnly=false) throw (decaf::lang::exceptions::NullPointerException)
*Creates a **IntArrayBuffer** (p. 1634) object that wraps the given array.*
- **IntArrayBuffer** (ByteArrayPerspective &array, std::size_t offset, std::size_t capacity, bool readOnly=false) throw (decaf::lang::exceptions::IndexOutOfBoundsException)
*Creates a byte buffer that wraps the passed **ByteArrayPerspective** (p. 843) and start at the given offset.*
- **IntArrayBuffer** (const IntArrayBuffer &other)
*Create a **IntArrayBuffer** (p. 1634) that mirrors this one, meaning it shares a reference to this buffers **ByteArrayPerspective** (p. 843) and when changes are made to that data it is reflected in both.*
- virtual ~**IntArrayBuffer** ()
- virtual int * **array** () throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException)
Returns the int array that backs this buffer (optional operation).
- virtual std::size_t **arrayOffset** () throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException)
Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).
- virtual IntBuffer * **asReadOnlyBuffer** () const
Creates a new, read-only int buffer that shares this buffer's content.
- virtual IntBuffer & **compact** () throw (decaf::nio::ReadOnlyBufferException)
Compacts this buffer.

- virtual `IntBuffer * duplicate ()`
Creates a new int buffer that shares this buffer's content.
- virtual `int get () throw (decaf::nio::BufferUnderflowException)`
Relative get method.
- virtual `int get (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)`
Absolute get method.
- virtual `bool hasArray () const`
Tells whether or not this buffer is backed by an accessible int array.
- virtual `bool isReadOnly () const`
Tells whether or not this buffer is read-only.
- virtual `IntBuffer & put (int value) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException)`
Writes the given doubles into this buffer at the current position, and then increments the position.
- virtual `IntBuffer & put (std::size_t index, int value) throw (lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException)`
Writes the given doubles into this buffer at the given index.
- virtual `IntBuffer * slice () const`
Creates a new IntBuffer whose content is a shared subsequence of this buffer's content.

Protected Member Functions

- virtual `void setReadOnly (bool value)`
*Sets this **ByteArrayBuffer** (p. 806) as Read-Only.*

6.337.1 Constructor & Destructor Documentation

6.337.1.1 `decaf::internal::nio::IntArrayBuffer::IntArrayBuffer (std::size_t capacity, bool readOnly = false)`

Creates a **IntArrayBuffer** (p. 1634) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.

The array is initially created with all elements initialized to zero.

Parameters

capacity - size of the array, this is the limit we read and write to.

readOnly - should this buffer be read-only, default as false

6.337.1.2 `decaf::internal::nio::IntArrayBuffer::IntArrayBuffer (int * array,
std::size_t offset, std::size_t capacity, bool readOnly = false)
throw (decaf::lang::exceptions::NullPointerException)`

Creates a **IntArrayBuffer** (p. 1634) object that wraps the given array.

If the own flag is set then it will delete this array when this object is deleted.

Parameters

array - array to wrap

offset - the position that is this buffers start pos.

capacity - size of the array, this is the limit we read and write to.

readOnly - should this buffer be read-only, default as false

Exceptions

NullPointerException if buffer is NULL

6.337.1.3 `decaf::internal::nio::IntArrayBuffer::IntArrayBuffer (
ByteArrayPerspective & array, std::size_t offset,
std::size_t capacity, bool readOnly = false) throw (
decaf::lang::exceptions::IndexOutOfBoundsException)`

Creates a byte buffer that wraps the passed **ByteArrayPerspective** (p. 843) and start at the given offset.

The capacity and limit of the new **IntArrayBuffer** (p. 1634) will be that of the remaining capacity of the passed buffer.

Parameters

array - the **ByteArrayPerspective** (p. 843) to wrap

offset - the offset into array where the buffer starts

capacity - the length of the array we are wrapping or limit.

readOnly - is this a readOnly buffer.

Exceptions

IndexOutOfBoundsException if offset is greater than array capacity.

6.337.1.4 `decaf::internal::nio::IntArrayBuffer::IntArrayBuffer (const
IntArrayBuffer & other)`

Create a **IntArrayBuffer** (p. 1634) that mirrors this one, meaning it shares a reference to this buffers **ByteArrayPerspective** (p. 843) and when changes are made to that data it is reflected in both.

Parameters

other - the **IntArrayBuffer** (p. 1634) this one is to mirror.

6.337.1.5 `virtual decaf::internal::nio::IntArrayBuffer::~~IntArrayBuffer ()`
[virtual]

6.337.2 Member Function Documentation

6.337.2.1 `virtual int* decaf::internal::nio::IntArrayBuffer::array ()`
`throw (decaf::lang::exceptions::UnsupportedOperationException,`
`decaf::nio::ReadOnlyBufferException)` [virtual]

Returns the int array that backs this buffer (optional operation).

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

the array that backs this Buffer

Exceptions

ReadOnlyBufferException if this Buffer is read only.

UnsupportedOperationException if the underlying store has no array.

Implements `decaf::nio::IntBuffer` (p. 1644).

6.337.2.2 `virtual std::size_t decaf::internal::nio::IntArrayBuffer::arrayOffset (`
`) throw (decaf::lang::exceptions::UnsupportedOperationException,`
`decaf::nio::ReadOnlyBufferException)` [virtual]

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

The offset into the backing array where index zero starts.

Exceptions

ReadOnlyBufferException if this Buffer is read only.

UnsupportedOperationException if the underlying store has no array.

Implements `decaf::nio::IntBuffer` (p. 1645).

6.337.2.3 `virtual IntBuffer* decaf::internal::nio::IntArrayBuffer::asReadOnlyBuffer`
`() const` [virtual]

Creates a new, read-only int buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow

the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns

The new, read-only int buffer which the caller then owns.

Implements **decaf::nio::IntBuffer** (p. 1645).

6.337.2.4 virtual IntBuffer& decaf::internal::nio::IntArrayBuffer::compact () throw (decaf::nio::ReadOnlyBufferException) [virtual]

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index $p = \text{position}()$ (p. 746) is copied to index zero, the byte at index $p + 1$ is copied to index one, and so forth until the byte at index $\text{limit}()$ (p. 745) - 1 is copied to index $n = \text{limit}()$ (p. 745) - 1 - p . The buffer's position is then set to $n+1$ and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns

a reference to this IntBuffer

Exceptions

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::IntBuffer** (p. 1645).

6.337.2.5 virtual IntBuffer* decaf::internal::nio::IntArrayBuffer::duplicate () [virtual]

Creates a new int buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

a new int Buffer which the caller owns.

Implements **decaf::nio::IntBuffer** (p. 1646).

6.337.2.6 `virtual int decaf::internal::nio::IntArrayBuffer::get () throw (decaf::nio::BufferUnderflowException) [virtual]`

Relative get method.

Reads the value at this buffer's current position, and then increments the position.

Returns

the int at the current position

Exceptions

BufferUnderflowException if there no more data to return

Implements `decaf::nio::IntBuffer` (p. 1648).

6.337.2.7 `virtual int decaf::internal::nio::IntArrayBuffer::get (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException) [virtual]`

Absolute get method.

Reads the value at the given index.

Parameters

index - the index in the Buffer where the int is to be read

Returns

the int that is located at the given index

Exceptions

IndexOutOfBoundsException - If index is not smaller than the buffer's limit

Implements `decaf::nio::IntBuffer` (p. 1647).

6.337.2.8 `virtual bool decaf::internal::nio::IntArrayBuffer::hasArray () const [inline, virtual]`

Tells whether or not this buffer is backed by an accessible int array.

If this method returns true then the `array` and `arrayOffset` methods may safely be invoked. Sub-classes should override this method if they do not have a backing array as this class always returns true.

Returns

true if, and only if, this buffer is backed by an array and is not read-only

Implements `decaf::nio::IntBuffer` (p. 1648).

6.337.2.9 `virtual bool decaf::internal::nio::IntArrayBuffer::isReadOnly () const [inline, virtual]`

Tells whether or not this buffer is read-only.

Returns

true if, and only if, this buffer is read-only

6.337.2.10 `virtual IntBuffer& decaf::internal::nio::IntArrayBuffer::put
(std::size_t index, int value) throw (
lang::exceptions::IndexOutOfBoundsException,
decaf::nio::ReadOnlyBufferException) [virtual]`

Writes the given doubles into this buffer at the given index.

Parameters

index - position in the Buffer to write the data

value - the doubles to write.

Returns

a reference to this buffer

Exceptions

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::IntBuffer** (p. 1649).

6.337.2.11 `virtual IntBuffer& decaf::internal::nio::IntArrayBuffer::put
(int value) throw (decaf::nio::BufferOverflowException,
decaf::nio::ReadOnlyBufferException) [virtual]`

Writes the given doubles into this buffer at the current position, and then increments the position.

Parameters

value - the doubles value to be written

Returns

a reference to this buffer

Exceptions

BufferOverflowException - If this buffer's current position is not smaller than its limit

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::IntBuffer** (p. 1649).

6.337.2.12 `virtual void decaf::internal::nio::IntArrayBuffer::setReadOnly (bool
value) [inline, protected, virtual]`

Sets this **ByteArrayBuffer** (p. 806) as Read-Only.

Parameters

value - true if this buffer is to be read-only.

6.337.2.13 `virtual IntBuffer* decaf::internal::nio::IntArrayBuffer::slice () const` [virtual]

Creates a new `IntBuffer` whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the newly create `IntBuffer` which the caller owns.

Implements `decaf::nio::IntBuffer` (p. 1651).

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/nio/IntArrayBuffer.h`

6.338 `decaf::nio::IntBuffer` Class Reference

This class defines four categories of operations upon int buffers:

```
#include <src/main/decaf/nio/IntBuffer.h>
```

Inheritance diagram for `decaf::nio::IntBuffer`:

Public Member Functions

- `virtual ~IntBuffer ()`
- `virtual std::string toString () const`
- `virtual int * array ()=0 throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException)`
Returns the int array that backs this buffer (optional operation).
- `virtual std::size_t arrayOffset ()=0 throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException)`
Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).
- `virtual IntBuffer * asReadOnlyBuffer () const =0`
Creates a new, read-only int buffer that shares this buffer's content.
- `virtual IntBuffer & compact ()=0 throw (ReadOnlyBufferException)`
Compacts this buffer.
- `virtual IntBuffer * duplicate ()=0`

Creates a new int buffer that shares this buffer's content.

- virtual int **get** ()=0 throw (BufferUnderflowException)
Relative get method.
- virtual int **get** (std::size_t index) const =0 throw (lang::exceptions::IndexOutOfBoundsException)
Absolute get method.
- **IntBuffer** & **get** (std::vector< int > buffer) throw (BufferUnderflowException)
Relative bulk get method.
- **IntBuffer** & **get** (int *buffer, std::size_t offset, std::size_t length) throw (BufferUnderflowException, lang::exceptions::NullPointerException)
Relative bulk get method.
- virtual bool **hasArray** () const =0
Tells whether or not this buffer is backed by an accessible int array.
- **IntBuffer** & **put** (**IntBuffer** &src) throw (BufferOverflowException, ReadOnlyBufferException, lang::exceptions::IllegalArgumentException)
This method transfers the ints remaining in the given source buffer into this buffer.
- **IntBuffer** & **put** (const int *buffer, std::size_t offset, std::size_t length) throw (BufferOverflowException, ReadOnlyBufferException, lang::exceptions::NullPointerException)
This method transfers ints into this buffer from the given source array.
- **IntBuffer** & **put** (std::vector< int > &buffer) throw (BufferOverflowException, ReadOnlyBufferException)
This method transfers the entire content of the given source ints array into this buffer.
- virtual **IntBuffer** & **put** (int value)=0 throw (BufferOverflowException, ReadOnlyBufferException)
Writes the given ints into this buffer at the current position, and then increments the position.
- virtual **IntBuffer** & **put** (std::size_t index, int value)=0 throw (lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException)
Writes the given ints into this buffer at the given index.
- virtual **IntBuffer** * **slice** () const =0
*Creates a new **IntBuffer** (p. 1641) whose content is a shared subsequence of this buffer's content.*
- virtual int **compareTo** (const **IntBuffer** &value) const
Compares this object with the specified object for order.
- virtual bool **equals** (const **IntBuffer** &value) const
- virtual bool **operator==** (const **IntBuffer** &value) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const **IntBuffer** &value) const
Compares this object to another and returns true if this object is considered to be less than the one passed.

Static Public Member Functions

- static **IntBuffer** * **allocate** (std::size_t capacity)
Allocates a new Double buffer.
- static **IntBuffer** * **wrap** (int *array, std::size_t offset, std::size_t length) throw (lang::exceptions::NullPointerException)
*Wraps the passed buffer with a new **IntBuffer** (p. 1641).*
- static **IntBuffer** * **wrap** (std::vector< int > &buffer)
*Wraps the passed STL int Vector in a **IntBuffer** (p. 1641).*

Protected Member Functions

- **IntBuffer** (std::size_t capacity)
*Creates a **IntBuffer** (p. 1641) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*

6.338.1 Detailed Description

This class defines four categories of operations upon int buffers: o Absolute and relative get and put methods that read and write single ints; o Relative bulk get methods that transfer contiguous sequences of ints from this buffer into an array; and o Relative bulk put methods that transfer contiguous sequences of ints from a int array or some other int buffer into this buffer o Methods for compacting, duplicating, and slicing a int buffer.

Double buffers can be created either by allocation, which allocates space for the buffer's content, by wrapping an existing int array into a buffer, or by creating a view of an existing byte buffer

Methods in this class that do not otherwise have a value to return are specified to return the buffer upon which they are invoked. This allows method invocations to be chained.

6.338.2 Constructor & Destructor Documentation

6.338.2.1 decaf::nio::IntBuffer::IntBuffer (std::size_t *capacity*) [protected]

Creates a **IntBuffer** (p. 1641) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.

The array is initially created with all elements initialized to zero.

Parameters

capacity - size and limit of the **Buffer** (p. 741) in doubles

6.338.2.2 virtual decaf::nio::IntBuffer::~~IntBuffer () [inline, virtual]

6.338.3 Member Function Documentation

6.338.3.1 static IntBuffer* decaf::nio::IntBuffer::allocate (std::size_t *capacity*)
[static]

Allocates a new Double buffer.

The new buffer's position will be zero, its limit will be its capacity, and its mark will be undefined. It will have a backing array, and its array offset will be zero.

Parameters

capacity - The size of the Double buffer in ints

Returns

the **IntBuffer** (p. 1641) that was allocated, caller owns.

6.338.3.2 virtual int* decaf::nio::IntBuffer::array () throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException) [pure virtual]

Returns the int array that backs this buffer (optional operation).

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

the array that backs this **Buffer** (p. 741)

Exceptions

ReadOnlyBufferException (p. 2520) if this **Buffer** (p. 741) is read only.

UnsupportedOperationException if the underlying store has no array.

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 1637).

6.338.3.3 virtual std::size_t decaf::nio::IntBuffer::arrayOffset () throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException) [pure virtual]

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

The offset into the backing array where index zero starts.

Exceptions

ReadOnlyBufferException (p. 2520) if this **Buffer** (p. 741) is read only.

UnsupportedOperationException if the underlying store has no array.

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 1637).

6.338.3.4 `virtual IntBuffer* decaf::nio::IntBuffer::asReadOnlyBuffer () const`
[pure virtual]

Creates a new, read-only int buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns

The new, read-only int buffer which the caller then owns.

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 1638).

6.338.3.5 `virtual IntBuffer& decaf::nio::IntBuffer::compact () throw (`
`ReadOnlyBufferException)` [pure virtual]

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index $p = \text{position}()$ (p. 746) is copied to index zero, the byte at index $p + 1$ is copied to index one, and so forth until the byte at index $\text{limit}()$ (p. 745) - 1 is copied to index $n = \text{limit}()$ (p. 745) - 1 - p . The buffer's position is then set to $n+1$ and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns

a reference to this **IntBuffer** (p. 1641)

Exceptions

ReadOnlyBufferException (p. 2520) - If this buffer is read-only

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 1638).

6.338.3.6 `virtual int decaf::nio::IntBuffer::compareTo (const IntBuffer & value)`
`const` [virtual]

Compares this object with the specified object for order.

Returns a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

Parameters

value - the Object to be compared.

Returns

a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

6.338.3.7 virtual IntBuffer* decaf::nio::IntBuffer::duplicate () [pure virtual]

Creates a new int buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

a new int **Buffer** (p. 741) which the caller owns.

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 1638).

6.338.3.8 virtual bool decaf::nio::IntBuffer::equals (const IntBuffer & value) const [virtual]

Returns

true if this value is considered equal to the passed value.

6.338.3.9 IntBuffer& decaf::nio::IntBuffer::get (std::vector< int > buffer) throw (BufferUnderflowException)

Relative bulk get method.

This method transfers values from this buffer into the given destination vector. An invocation of this method of the form `src.get(a)` behaves in exactly the same way as the invocation. The vector must be sized to the amount of data that is to be read, that is to say, the caller should call `buffer.resize(N)` before calling this get method.

Returns

a reference to this **Buffer** (p. 741)

Exceptions

BufferUnderflowException (p. 774) - If there are fewer than length ints remaining in this buffer

6.338.3.10 `virtual int decaf::nio::IntBuffer::get (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)` [pure virtual]

Absolute get method.

Reads the value at the given index.

Parameters

index - the index in the **Buffer** (p. 741) where the int is to be read

Returns

the int that is located at the given index

Exceptions

IndexOutOfBoundsException - If index is not smaller than the buffer's limit

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 1639).

6.338.3.11 `IntBuffer& decaf::nio::IntBuffer::get (int * buffer, std::size_t offset, std::size_t length) throw (BufferUnderflowException, lang::exceptions::NullPointerException)`

Relative bulk get method.

This method transfers ints from this buffer into the given destination array. If there are fewer ints remaining in the buffer than are required to satisfy the request, that is, if `length > remaining()` (p. 746), then no bytes are transferred and a **BufferUnderflowException** (p. 774) is thrown.

Otherwise, this method copies `length` ints from this buffer into the given array, starting at the current position of this buffer and at the given offset in the array. The position of this buffer is then incremented by `length`.

Parameters

buffer - pointer to an allocated buffer to fill

offset - position in the buffer to start filling

length - amount of data to put in the passed buffer

Returns

a reference to this **Buffer** (p. 741)

Exceptions

BufferUnderflowException (p. 774) - If there are fewer than `length` ints remaining in this buffer

NullPointerException if the passed buffer is null.

6.338.3.12 `virtual int decaf::nio::IntBuffer::get () throw (BufferUnderflowException)` [pure virtual]

Relative get method.

Reads the value at this buffer's current position, and then increments the position.

Returns

the int at the current position

Exceptions

BufferUnderflowException (p. 774) if there no more data to return

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 1639).

6.338.3.13 virtual bool decaf::nio::IntBuffer::hasArray () const [pure virtual]

Tells whether or not this buffer is backed by an accessible int array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Sub-classes should override this method if they do not have a backing array as this class always returns true.

Returns

true if, and only if, this buffer is backed by an array and is not read-only

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 1639).

6.338.3.14 virtual bool decaf::nio::IntBuffer::operator< (const IntBuffer & value) const [virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

Parameters

value - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

6.338.3.15 virtual bool decaf::nio::IntBuffer::operator== (const IntBuffer & value) const [virtual]

Compares equality between this object and the one passed.

Parameters

value - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

6.338.3.16 `virtual IntBuffer& decaf::nio::IntBuffer::put (int value) throw (BufferOverflowException, ReadOnlyBufferException) [pure virtual]`

Writes the given ints into this buffer at the current position, and then increments the position.

Parameters

value - the ints value to be written

Returns

a reference to this buffer

Exceptions

BufferOverflowException (p. 771) - If this buffer's current position is not smaller than its limit

ReadOnlyBufferException (p. 2520) - If this buffer is read-only

Implemented in `decaf::internal::nio::IntArrayBuffer` (p. 1640).

6.338.3.17 `virtual IntBuffer& decaf::nio::IntBuffer::put (std::size_t index, int value) throw (lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException) [pure virtual]`

Writes the given ints into this buffer at the given index.

Parameters

index - position in the **Buffer** (p. 741) to write the data

value - the ints to write.

Returns

a reference to this buffer

Exceptions

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException (p. 2520) - If this buffer is read-only

Implemented in `decaf::internal::nio::IntArrayBuffer` (p. 1640).

6.338.3.18 `IntBuffer& decaf::nio::IntBuffer::put (const int * buffer, std::size_t offset, std::size_t length) throw (BufferOverflowException, ReadOnlyBufferException, lang::exceptions::NullPointerException)`

This method transfers ints into this buffer from the given source array.

If there are more ints to be copied from the array than remain in this buffer, that is, if `length > remaining()` (p. 746), then no ints are transferred and a **BufferOverflowException** (p. 771) is thrown.

Otherwise, this method copies `length` bytes from the given array into this buffer, starting at the given `offset` in the array and at the current position of this buffer. The position of this buffer is then incremented by `length`.

Parameters

buffer- The array from which ints are to be read
offset- The offset within the array of the first int to be read;
length - The number of ints to be read from the given array

Returns

a reference to this buffer

Exceptions

BufferOverflowException (p. 771) - If there is insufficient space in this buffer
ReadOnlyBufferException (p. 2520) - If this buffer is read-only
NullPointerException if the passed buffer is null.

6.338.3.19 IntBuffer& decaf::nio::IntBuffer::put (std::vector< int > & *buffer*) throw (BufferOverflowException, ReadOnlyBufferException)

This method transfers the entire content of the given source ints array into this buffer.

This is the same as calling put(&buffer[0], 0, buffer.size())

Parameters

buffer - The buffer whose contents are copied to this **IntBuffer** (p. 1641)

Returns

a reference to this buffer

Exceptions

BufferOverflowException (p. 771) - If there is insufficient space in this buffer
ReadOnlyBufferException (p. 2520) - If this buffer is read-only

6.338.3.20 IntBuffer& decaf::nio::IntBuffer::put (IntBuffer & *src*) throw (BufferOverflowException, ReadOnlyBufferException, lang::exceptions::IllegalArgumentException)

This method transfers the ints remaining in the given source buffer into this buffer.

If there are more ints remaining in the source buffer than in this buffer, that is, if src.remaining() > remaining() (p. 746), then no ints are transferred and a **BufferOverflowException** (p. 771) is thrown.

Otherwise, this method copies n = src.remaining() ints from the given buffer into this buffer, starting at each buffer's current position. The positions of both buffers are then incremented by n.

Parameters

src - the buffer to take ints from an place in this one.

Returns

a reference to this buffer

Exceptions

BufferOverflowException (p. 771) - If there is insufficient space in this buffer for the remaining ints in the source buffer

IllegalArgumentException - If the source buffer is this buffer

ReadOnlyBufferException (p. 2520) - If this buffer is read-only

6.338.3.21 virtual IntBuffer* decaf::nio::IntBuffer::slice () const [pure virtual]

Creates a new **IntBuffer** (p. 1641) whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the newly create **IntBuffer** (p. 1641) which the caller owns.

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 1641).

6.338.3.22 virtual std::string decaf::nio::IntBuffer::toString () const [virtual]**Returns**

a std::string describing this object

6.338.3.23 static IntBuffer* decaf::nio::IntBuffer::wrap (int * array, std::size_t offset, std::size_t length) throw (lang::exceptions::NullPointerException) [static]

Wraps the passed buffer with a new **IntBuffer** (p. 1641).

The new buffer will be backed by the given int array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be array.length, its position will be offset, its limit will be offset + length, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

array - The array that will back the new buffer

offset - The offset of the subarray to be used

length - The length of the subarray to be used

Returns

a new **IntBuffer** (p. 1641) that is backed by buffer, caller owns.

6.338.3.24 static **IntBuffer*** decaf::nio::IntBuffer::wrap (**std::vector< int > &buffer**) [static]

Wraps the passed STL int Vector in a **IntBuffer** (p. 1641).

The new buffer will be backed by the given int array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be `buffer.size()`, its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

buffer - The vector that will back the new buffer, the vector must have been sized to the desired size already by calling `vector.resize(N)`.

Returns

a new **IntBuffer** (p. 1641) that is backed by `buffer`, caller owns.

The documentation for this class was generated from the following file:

- `src/main/decaf/nio/IntBuffer.h`

6.339 decaf::lang::Integer Class Reference

```
#include <src/main/decaf/lang/Integer.h>
```

Inheritance diagram for `decaf::lang::Integer`:

Public Member Functions

- **Integer** (int value)
- **Integer** (const **std::string** &value) throw (**exceptions::NumberFormatException**)
- virtual **~Integer** ()
- virtual int **compareTo** (const **Integer** &i) const
*Compares this **Integer** (p. 1652) instance with another.*
- bool **equals** (const **Integer** &i) const
- virtual bool **operator==** (const **Integer** &i) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const **Integer** &i) const
Compares this object to another and returns true if this object is considered to be less than the one passed.
- virtual int **compareTo** (const int &i) const
*Compares this **Integer** (p. 1652) instance with another.*
- bool **equals** (const int &i) const
- virtual bool **operator==** (const int &i) const

Compares equality between this object and the one passed.

- virtual bool **operator<** (const int &i) const
Compares this object to another and returns true if this object is considered to be less than the one passed.
- std::string **toString** () const
- virtual double **doubleValue** () const
Answers the double value which the receiver represents.
- virtual float **floatValue** () const
Answers the float value which the receiver represents.
- virtual unsigned char **byteValue** () const
Answers the byte value which the receiver represents.
- virtual short **shortValue** () const
Answers the short value which the receiver represents.
- virtual int **intValue** () const
Answers the int value which the receiver represents.
- virtual long long **longValue** () const
Answers the long value which the receiver represents.

Static Public Member Functions

- static **Integer decode** (const std::string &value) throw (exceptions::NumberFormatException)
*Decodes a String into a **Integer** (p. 1652).*
- static int **reverseBytes** (int value)
Returns the value obtained by reversing the order of the bytes in the two's complement representation of the specified int value.
- static int **reverse** (int value)
Returns the value obtained by reversing the order of the bits in the two's complement binary representation of the specified int value.
- static int **parseInt** (const std::string &s, int radix) throw (exceptions::NumberFormatException)
Parses the string argument as a signed int in the radix specified by the second argument.
- static int **parseInt** (const std::string &s) throw (exceptions::NumberFormatException)
Parses the string argument as a signed decimal int.
- static **Integer valueOf** (int value)
*Returns a **Integer** (p. 1652) instance representing the specified int value.*

- static **Integer** **valueOf** (const std::string &value) throw (exceptions::NumberFormatException)
*Returns a **Integer** (p. 1652) object holding the value given by the specified std::string.*
- static **Integer** **valueOf** (const std::string &value, int radix) throw (exceptions::NumberFormatException)
*Returns a **Integer** (p. 1652) object holding the value extracted from the specified std::string when parsed with the radix given by the second argument.*
- static int **bitCount** (int value)
Returns the number of one-bits in the two's complement binary representation of the specified int value.
- static std::string **toString** (int value)
Converts the int to a String representation.
- static std::string **toString** (int value, int radix)
Returns a string representation of the first argument in the radix specified by the second argument.
- static std::string **toHexString** (int value)
Returns a string representation of the integer argument as an unsigned integer in base 16.
- static std::string **toOctalString** (int value)
Returns a string representation of the integer argument as an unsigned integer in base 8.
- static std::string **toBinaryString** (int value)
Returns a string representation of the integer argument as an unsigned integer in base 2.
- static int **highestOneBit** (int value)
Returns an int value with at most a single one-bit, in the position of the highest-order ("leftmost") one-bit in the specified int value.
- static int **lowestOneBit** (int value)
Returns an int value with at most a single one-bit, in the position of the lowest-order ("rightmost") one-bit in the specified int value.
- static int **numberOfLeadingZeros** (int value)
Returns the number of zero bits preceding the highest-order ("leftmost") one-bit in the two's complement binary representation of the specified int value.
- static int **numberOfTrailingZeros** (int value)
Returns the number of zero bits following the lowest-order ("rightmost") one-bit in the two's complement binary representation of the specified int value.
- static int **rotateLeft** (int value, int distance)
Returns the value obtained by rotating the two's complement binary representation of the specified int value left by the specified number of bits.
- static int **rotateRight** (int value, int distance)
Returns the value obtained by rotating the two's complement binary representation of the specified int value right by the specified number of bits.

- static int **signum** (int value)
Returns the signum function of the specified int value.

Static Public Attributes

- static const int **SIZE** = 32
The size in bits of the primitive int type.
- static const int **MAX_VALUE** = (int)0x7FFFFFFF
The maximum value that the primitive type can hold.
- static const int **MIN_VALUE** = (int)0x80000000
The minimum value that the primitive type can hold.

6.339.1 Constructor & Destructor Documentation

6.339.1.1 decaf::lang::Integer::Integer (int value)

Parameters

value The primitive value to wrap in an Integer (p. 1652) instance.

6.339.1.2 decaf::lang::Integer::Integer (const std::string & value) throw (exceptions::NumberFormatException)

Parameters

value The base 10 encoded string to decode to an Integer (p. 1652) and wrap.

Exceptions

NumberFormatException

6.339.1.3 virtual decaf::lang::Integer::~~Integer () [inline, virtual]

6.339.2 Member Function Documentation

6.339.2.1 static int decaf::lang::Integer::bitCount (int value) [static]

Returns the number of one-bits in the two's complement binary representation of the specified int value.

This function is sometimes referred to as the population count.

Parameters

value - the int to count

Returns

the number of one-bits in the two's complement binary representation of the specified int value.

6.339.2.2 `virtual unsigned char decaf::lang::Integer::byteValue () const [inline, virtual]`

Answers the byte value which the receiver represents.

Returns

int the value of the receiver.

6.339.2.3 `virtual int decaf::lang::Integer::compareTo (const int & i) const [virtual]`

Compares this **Integer** (p.1652) instance with another.

Parameters

i - the **Integer** (p.1652) instance to be compared

Returns

zero if this object represents the same integer value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

Implements **decaf::lang::Comparable< int >** (p.1010).

6.339.2.4 `virtual int decaf::lang::Integer::compareTo (const Integer & i) const [virtual]`

Compares this **Integer** (p.1652) instance with another.

Parameters

i - the **Integer** (p.1652) instance to be compared

Returns

zero if this object represents the same integer value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

6.339.2.5 `static Integer decaf::lang::Integer::decode (const std::string & value) throw (exceptions::NumberFormatException) [static]`

Decodes a String into a **Integer** (p.1652).

Accepts decimal, hexadecimal, and octal numbers given by the following grammar:

The sequence of characters following an (optional) negative sign and/or radix specifier ("0x", "0X", "#", or leading zero) is parsed as by the `Integer.parseInt` method with the indicated radix (10, 16, or 8). This sequence of characters must represent a positive value or a `NumberFormatException` will be thrown. The result is negated if first character of the specified String is the minus sign. No whitespace characters are permitted in the string.

Parameters

value - The string to decode

Returns

a **Integer** (p. 1652) object containing the decoded value

Exceptions

NumberFormatException if the string is not formatted correctly.

6.339.2.6 `virtual double decaf::lang::Integer::doubleValue () const [inline, virtual]`

Answers the double value which the receiver represents.

Returns

double the value of the receiver.

6.339.2.7 `bool decaf::lang::Integer::equals (const Integer & i) const [inline]`

Parameters

i - the **Integer** (p. 1652) object to compare against.

Returns

true if the two **Integer** (p. 1652) Objects have the same value.

6.339.2.8 `bool decaf::lang::Integer::equals (const int & i) const [inline, virtual]`

Parameters

i - the **Integer** (p. 1652) object to compare against.

Returns

true if the two **Integer** (p. 1652) Objects have the same value.

Implements `decaf::lang::Comparable< int >` (p. 1010).

6.339.2.9 `virtual float decaf::lang::Integer::floatValue () const [inline, virtual]`

Answers the float value which the receiver represents.

Returns

float the value of the receiver.

6.339.2.10 `static int decaf::lang::Integer::highestOneBit (int value) [static]`

Returns an int value with at most a single one-bit, in the position of the highest-order ("leftmost") one-bit in the specified int value.

Returns zero if the specified value has no one-bits in its two's complement binary representation, that is, if it is equal to zero.

Parameters

value - the int to be inspected

Returns

an int value with a single one-bit, in the position of the highest-order one-bit in the specified value, or zero if the specified value is itself equal to zero.

6.339.2.11 `virtual int decaf::lang::Integer::intValue () const [inline, virtual]`

Answers the int value which the receiver represents.

Returns

int the value of the receiver.

6.339.2.12 `virtual long long decaf::lang::Integer::longValue () const [inline, virtual]`

Answers the long value which the receiver represents.

Returns

long the value of the receiver.

6.339.2.13 `static int decaf::lang::Integer::lowestOneBit (int value) [static]`

Returns an int value with at most a single one-bit, in the position of the lowest-order ("rightmost") one-bit in the specified int value.

Returns zero if the specified value has no one-bits in its two's complement binary representation, that is, if it is equal to zero.

Parameters

value - the int to be inspected

Returns

an int value with a single one-bit, in the position of the lowest-order one-bit in the specified value, or zero if the specified value is itself equal to zero.

6.339.2.14 `static int decaf::lang::Integer::numberOfLeadingZeros (int value)`
[static]

Returns the number of zero bits preceding the highest-order ("leftmost") one-bit in the two's complement binary representation of the specified int value.

Returns 32 if the specified value has no one-bits in its two's complement representation, in other words if it is equal to zero.

Note that this method is closely related to the logarithm base 2. For all positive int values x:

$\ast \text{floor}(\log_2(x)) = 31 - \text{numberOfLeadingZeros}(x)$ $\ast \text{ceil}(\log_2(x)) = 32 - \text{numberOfLeadingZeros}(x - 1)$

Parameters

value - the int to be inspected

Returns

the number of zero bits preceding the highest-order ("leftmost") one-bit in the two's complement binary representation of the specified int value, or 32 if the value is equal to zero.

6.339.2.15 `static int decaf::lang::Integer::numberOfTrailingZeros (int value)`
[static]

Returns the number of zero bits following the lowest-order ("rightmost") one-bit in the two's complement binary representation of the specified int value.

Returns 32 if the specified value has no one-bits in its two's complement representation, in other words if it is equal to zero.

Parameters

value - the int to be inspected

Returns

the number of zero bits following the lowest-order ("rightmost") one-bit in the two's complement binary representation of the specified int value, or 32 if the value is equal to zero.

6.339.2.16 `virtual bool decaf::lang::Integer::operator< (const int & i) const`
[inline, virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

Parameters

i - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable**< **int** > (p.1011).

6.339.2.17 **virtual bool decaf::lang::Integer::operator< (const Integer & i) const**
[inline, virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

Parameters

i - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

6.339.2.18 **virtual bool decaf::lang::Integer::operator== (const Integer & i) const**
[inline, virtual]

Compares equality between this object and the one passed.

Parameters

i - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

6.339.2.19 **virtual bool decaf::lang::Integer::operator== (const int & i) const**
[inline, virtual]

Compares equality between this object and the one passed.

Parameters

i - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable**< **int** > (p.1011).

6.339.2.20 `static int decaf::lang::Integer::parseInt (const std::string & s, int radix) throw (exceptions::NumberFormatException) [static]`

Parses the string argument as a signed int in the radix specified by the second argument.

The characters in the string must all be digits, of the specified radix (as determined by whether **Character.digit(char, int)** (p. 917) returns a nonnegative value) except that the first character may be an ASCII minus sign '-' to indicate a negative value. The resulting byte value is returned.

An exception of type `NumberFormatException` is thrown if any of the following situations occurs:

- * The first argument is null or is a string of length zero.
- * The radix is either smaller than **Character.MIN_RADIX** (p. 921) or larger than **Character.MAX_RADIX** (p. 921).
- * Any character of the string is not a digit of the specified radix, except that the first character may be a minus sign '-' provided that the string is longer than length 1.
- * The value represented by the string is not a value of type int.

Parameters

s - the String containing the int representation to be parsed

radix - the radix to be used while parsing s

Returns

the int represented by the string argument in the specified radix.

Exceptions

NumberFormatException - If String does not contain a parsable int.

6.339.2.21 `static int decaf::lang::Integer::parseInt (const std::string & s) throw (exceptions::NumberFormatException) [static]`

Parses the string argument as a signed decimal int.

The characters in the string must all be decimal digits, except that the first character may be an ASCII minus sign '-' to indicate a negative value. The resulting int value is returned, exactly as if the argument and the radix 10 were given as arguments to the `parseInt(const std::string, int)` method.

Parameters

s - String to convert to a int

Returns

the converted int value

Exceptions

NumberFormatException if the string is not a int.

6.339.2.22 `static int decaf::lang::Integer::reverse (int value) [static]`

Returns the value obtained by reversing the order of the bits in the two's complement binary representation of the specified int value.

Parameters

value - the value whose bits are to be reversed

Returns

the reversed bits int.

6.339.2.23 static int decaf::lang::Integer::reverseBytes (int *value*) [static]

Returns the value obtained by reversing the order of the bytes in the two's complement representation of the specified int value.

Parameters

value - the int whose bytes we are to reverse

Returns

the reversed int.

6.339.2.24 static int decaf::lang::Integer::rotateLeft (int *value*, int *distance*) [static]

Returns the value obtained by rotating the two's complement binary representation of the specified int value left by the specified number of bits.

(Bits shifted out of the left hand, or high-order, side reenter on the right, or low-order.)

Note that left rotation with a negative distance is equivalent to right rotation: rotateLeft(val, -distance) == rotateRight(val, distance). Note also that rotation by any multiple of 32 is a no-op, so all but the last five bits of the rotation distance can be ignored, even if the distance is negative: rotateLeft(val, distance) == rotateLeft(val, distance & 0x1F).

Parameters

value - the int to be inspected

distance - the number of bits to rotate

Returns

the value obtained by rotating the two's complement binary representation of the specified int value left by the specified number of bits.

6.339.2.25 static int decaf::lang::Integer::rotateRight (int *value*, int *distance*) [static]

Returns the value obtained by rotating the two's complement binary representation of the specified int value right by the specified number of bits.

(Bits shifted out of the right hand, or low-order, side reenter on the left, or high-order.)

Note that right rotation with a negative distance is equivalent to left rotation: rotateRight(val, -distance) == rotateLeft(val, distance). Note also that rotation by any multiple of 32 is a no-op, so all but the last five bits of the rotation distance can be ignored, even if the distance is negative: rotateRight(val, distance) == rotateRight(val, distance & 0x1F).

Parameters

value - the int to be inspected

distance - the number of bits to rotate

Returns

the value obtained by rotating the two's complement binary representation of the specified int value right by the specified number of bits.

6.339.2.26 `virtual short decaf::lang::Integer::shortValue () const [inline, virtual]`

Answers the short value which the receiver represents.

Returns

int the value of the receiver.

6.339.2.27 `static int decaf::lang::Integer::signum (int value) [static]`

Returns the signum function of the specified int value.

(The return value is -1 if the specified value is negative; 0 if the specified value is zero; and 1 if the specified value is positive.)

Parameters

value - the int to be inspected

Returns

the signum function of the specified int value.

6.339.2.28 `static std::string decaf::lang::Integer::toBinaryString (int value) [static]`

Returns a string representation of the integer argument as an unsigned integer in base 2.

The unsigned integer value is the argument plus 2^{32} if the argument is negative; otherwise it is equal to the argument. This value is converted to a string of ASCII digits in binary (base 2) with no extra leading 0s. If the unsigned magnitude is zero, it is represented by a single zero character '0'; otherwise, the first character of the representation of the unsigned magnitude will not be the zero character.

The characters '0' and '1' are used as binary digits.

Parameters

value - the int to be translated to a binary string

Returns

the unsigned int value as a binary string

6.339.2.29 `static std::string decaf::lang::Integer::toHexString (int value)`
 [static]

Returns a string representation of the integer argument as an unsigned integer in base 16.

The unsigned integer value is the argument plus 2^{32} if the argument is negative; otherwise, it is equal to the argument. This value is converted to a string of ASCII digits in hexadecimal (base 16) with no extra leading 0s. If the unsigned magnitude is zero, it is represented by a single zero character '0'; otherwise, the first character of the representation of the unsigned magnitude will not be the zero character. The following characters are used as hexadecimal digits:

0123456789abcdef

If uppercase letters are desired, the `toUpperCase()` method may be called on the result:

Parameters

value - the int to be translated to an Octal string

Returns

the unsigned int value as a Octal string

6.339.2.30 `static std::string decaf::lang::Integer::toOctalString (int value)`
 [static]

Returns a string representation of the integer argument as an unsigned integer in base 8.

The unsigned integer value is the argument plus 2^{32} if the argument is negative; otherwise, it is equal to the argument. This value is converted to a string of ASCII digits in octal (base 8) with no extra leading 0s.

If the unsigned magnitude is zero, it is represented by a single zero character '0'; otherwise, the first character of the representation of the unsigned magnitude will not be the zero character. The following characters are used as octal digits:

01234567

Parameters

value - the int to be translated to an Octal string

Returns

the unsigned int value as a Octal string

6.339.2.31 `std::string decaf::lang::Integer::toString () const`**Returns**

this Integer (p. 1652) Object as a String Representation

6.339.2.32 `static std::string decaf::lang::Integer::toString (int value, int radix)`
 [static]

Returns a string representation of the first argument in the radix specified by the second argument.

If the radix is smaller than **Character.MIN_RADIX** (p.921) or larger than **Character.MAX_RADIX** (p.921), then the radix 10 is used instead.

If the first argument is negative, the first element of the result is the ASCII minus character '-'. If the first argument is not negative, no sign character appears in the result.

The remaining characters of the result represent the magnitude of the first argument. If the magnitude is zero, it is represented by a single zero character '0'; otherwise, the first character of the representation of the magnitude will not be the zero character. The following ASCII characters are used as digits:

0123456789abcdefghijklmnopqrstuvwxyz

Parameters

value - the int to convert to a string
radix - the radix to format the string in

Returns

an int formatted to the string value of the radix given.

6.339.2.33 static std::string decaf::lang::Integer::toString (int *value*) [static]

Converts the int to a String representation.

Parameters

value The int to convert to a std::string instance.

Returns

string representation

6.339.2.34 static Integer decaf::lang::Integer::valueOf (const std::string & *value*, int *radix*) throw (exceptions::NumberFormatException) [static]

Returns a **Integer** (p.1652) object holding the value extracted from the specified std::string when parsed with the radix given by the second argument.

The first argument is interpreted as representing a signed int in the radix specified by the second argument, exactly as if the argument were given to the parseInt(std::string, int) method. The result is a **Integer** (p.1652) object that represents the int value specified by the string.

Parameters

value - std::string to parse as base (radix)
radix - base of the string to parse.

Returns

new **Integer** (p.1652) Object wrapping the primitive

Exceptions

NumberFormatException if the string is not a valid int.

6.339.2.35 `static Integer decaf::lang::Integer::valueOf (int value) [inline, static]`

Returns a **Integer** (p.1652) instance representing the specified int value.

Parameters

value - the int to wrap

Returns

the new **Integer** (p.1652) object wrapping value.

6.339.2.36 `static Integer decaf::lang::Integer::valueOf (const std::string & value) throw (exceptions::NumberFormatException) [static]`

Returns a **Integer** (p.1652) object holding the value given by the specified std::string.

The argument is interpreted as representing a signed decimal int, exactly as if the argument were given to the `parseInt(std::string)` method. The result is a **Integer** (p.1652) object that represents the int value specified by the string.

Parameters

value - std::string to parse as base 10

Returns

new **Integer** (p.1652) Object wrapping the primitive

Exceptions

NumberFormatException if the string is not a decimal int.

6.339.3 Field Documentation

6.339.3.1 `const int decaf::lang::Integer::MAX_VALUE = (int)0x7FFFFFFF [static]`

The maximum value that the primitive type can hold.

6.339.3.2 `const int decaf::lang::Integer::MIN_VALUE = (int)0x80000000 [static]`

The minimum value that the primitive type can hold.

6.339.3.3 `const int decaf::lang::Integer::SIZE = 32 [static]`

The size in bits of the primitive int type.

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Integer.h`

6.340 activemq::commands::IntegerResponse Class Reference

```
#include <src/main/activemq/commands/IntegerResponse.h>
```

Inheritance diagram for activemq::commands::IntegerResponse:

Public Member Functions

- **IntegerResponse** ()
- virtual **~IntegerResponse** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **IntegerResponse * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1372) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent.*
- virtual int **getResult** () const
- virtual void **setResult** (int result)

Static Public Attributes

- static const unsigned char **ID_INTEGERRESPONSE** = 34

Protected Member Functions

- **IntegerResponse** (const **IntegerResponse** &)
- **IntegerResponse & operator=** (const **IntegerResponse** &)

Protected Attributes

- int **result**

6.340.1 Constructor & Destructor Documentation

6.340.1.1 `activemq::commands::IntegerResponse::IntegerResponse (const IntegerResponse &) [inline, protected]`

6.340.1.2 `activemq::commands::IntegerResponse::IntegerResponse ()`

6.340.1.3 `virtual activemq::commands::IntegerResponse::~~IntegerResponse () [virtual]`

6.340.2 Member Function Documentation

6.340.2.1 `virtual IntegerResponse* activemq::commands::IntegerResponse::cloneDataStructure () const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Reimplemented from `activemq::commands::Response` (p.2612).

6.340.2.2 `virtual void activemq::commands::IntegerResponse::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

Reimplemented from `activemq::commands::Response` (p.2612).

6.340.2.3 `virtual bool activemq::commands::IntegerResponse::equals (const DataStructure * value) const [virtual]`

Compares the `DataStructure` (p.1372) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if `DataStructure`'s are Equal.

Reimplemented from `activemq::commands::Response` (p.2613).

6.340.2.4 `virtual unsigned char activemq::commands::IntegerResponse::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataSet** (p. 1372) type copy.

Reimplemented from **activemq::commands::Response** (p. 2613).

6.340.2.5 `virtual int activemq::commands::IntegerResponse::getResult () const`
[virtual]

6.340.2.6 `IntegerResponse& activemq::commands::IntegerResponse::operator= (`
`const IntegerResponse &)` [inline, protected]

6.340.2.7 `virtual void activemq::commands::IntegerResponse::setResult (int`
`result)` [virtual]

6.340.2.8 `virtual std::string activemq::commands::IntegerResponse::toString ()`
`const` [virtual]

Returns a string containing the information for this **DataSet** (p. 1372) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::Response** (p. 2614).

6.340.3 Field Documentation

6.340.3.1 `const unsigned char activemq::commands::IntegerResponse::ID_ -`
`INTEGERRESPONSE = 34` [static]

6.340.3.2 `int activemq::commands::IntegerResponse::result` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/IntegerResponse.h`

**6.341 activemq::wireformat::openwire::marshal::v2::IntegerResponseMar
Class Reference**

Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 1669).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/IntegerResponseMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller**:

Public Member Functions

- **IntegerResponseMarshaller** ()
- virtual **~IntegerResponseMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.341.1 Detailed Description

Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p.1669). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.341.2 Constructor & Destructor Documentation

6.341.2.1 `activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller::IntegerResponseMarshaller() [inline]`

6.341.2.2 `virtual activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller::~~IntegerResponseMarshaller() [inline, virtual]`

6.341.3 Member Function Documentation

6.341.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller::createObject() const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 2621).

6.341.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller::getDataStructureType() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 2621).

6.341.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 2621).

6.341.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 2622).

6.341.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 2622).

6.341.3.6 `virtual void activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 2623).

6.341.3.7 `virtual void activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller` (p. 2623).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/IntegerResponseMarshaller.h`

6.342 `activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller` Class Reference

Marshaling code for Open Wire Format for `IntegerResponseMarshaller` (p. 1673).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/IntegerResponseMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller:

Public Member Functions

- **IntegerResponseMarshaller** ()
- virtual **~IntegerResponseMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.342.1 Detailed Description

Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p.1673). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.342.2 Constructor & Destructor Documentation

6.342.2.1 `activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller::IntegerResponseMarshaller() [inline]`

6.342.2.2 `virtual activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller::~~IntegerResponseMarshaller() [inline, virtual]`

6.342.3 Member Function Documentation

6.342.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller::createObject() const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 2634).

6.342.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller::getDataStructureType() const [virtual]`

Get the Data Structure Type that identifies this Marshaller.

Returns

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 2635).

6.342.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 2635).

6.342.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 2635).

6.342.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 2636).

6.342.3.6 `virtual void activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 2636).

6.342.3.7 `virtual void activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` (p. 2637).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/IntegerResponseMarshaller.h`

6.343 `activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller` Class Reference

Marshaling code for Open Wire Format for `IntegerResponseMarshaller` (p. 1677).


```
#include <src/main/activemq/wireformat/openwire/marshal/v3/IntegerResponseMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller:

Public Member Functions

- **IntegerResponseMarshaller** ()
- virtual **~IntegerResponseMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.343.1 Detailed Description

Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p.1677). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.343.2 Constructor & Destructor Documentation

6.343.2.1 `activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller::IntegerResponseMarshaller() [inline]`

6.343.2.2 `virtual activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller::~~IntegerResponseMarshaller() [inline, virtual]`

6.343.3 Member Function Documentation

6.343.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller::createObject() const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 2625).

6.343.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller::getDataStructureType() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 2626).

6.343.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 2626).

6.343.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 2626).

6.343.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 2627).

6.343.3.6 `virtual void activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 2627).

6.343.3.7 `virtual void activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::ResponseMarshaller` (p. 2628).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/IntegerResponseMarshaller.h`

6.344 `activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller` Class Reference

Marshaling code for Open Wire Format for `IntegerResponseMarshaller` (p. 1681).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/IntegerResponseMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller:

Public Member Functions

- **IntegerResponseMarshaller** ()
- virtual **~IntegerResponseMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.344.1 Detailed Description

Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p.1681). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.344.2 Constructor & Destructor Documentation

6.344.2.1 `activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller::IntegerResponseMarshaller() [inline]`

6.344.2.2 `virtual activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller::~~IntegerResponseMarshaller() [inline, virtual]`

6.344.3 Member Function Documentation

6.344.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller::createObject() const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 2639).

6.344.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller::getDataStructureType() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 2639).

6.344.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 2639).

6.344.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 2640).

6.344.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller** (p. 2640).

6.344.3.6 `virtual void activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 2641).

6.344.3.7 `virtual void activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::ResponseMarshaller` (p. 2641).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/IntegerResponseMarshaller.h`

6.345 `activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller` Class Reference

Marshaling code for Open Wire Format for `IntegerResponseMarshaller` (p. 1685).


```
#include <src/main/activemq/wireformat/openwire/marshal/v5/IntegerResponseMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller:

Public Member Functions

- **IntegerResponseMarshaller** ()
- virtual **~IntegerResponseMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.345.1 Detailed Description

Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p.1685). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.345.2 Constructor & Destructor Documentation

6.345.2.1 `activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller::IntegerResponseMarshaller() [inline]`

6.345.2.2 `virtual activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller::~~IntegerResponseMarshaller() [inline, virtual]`

6.345.3 Member Function Documentation

6.345.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller::createObject() const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 2630).

6.345.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller::getDataStructureType() const [virtual]`

Get the Data Structure Type that identifies this Marshaller.

Returns

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` (p. 2630).

6.345.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 2630).

6.345.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 2631).

6.345.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 2631).

```

6.345.3.6 virtual void ac-
    tivemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller::tightMarshal2
    ( OpenWireFormat * wireFormat, commands::DataStructure
    * dataStructure, decaf::io::DataOutputStream * dataOut,
    utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 2632).

```

6.345.3.7 virtual void ac-
    tivemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller::tightUnmarshal
    ( OpenWireFormat * wireFormat, commands::DataStructure
    * dataStructure, decaf::io::DataInputStream * dataIn,
    utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]

```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller** (p. 2632).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**IntegerResponseMarshaller.h**

6.346 activemq::transport::mock::InternalCommandListener Class Reference

Listens for Commands sent from the **MockTransport** (p. 2227).

```
#include <src/main/activemq/transport/mock/InternalCommandListener.h>
```

Inheritance diagram for `activemq::transport::mock::InternalCommandListener`:

Public Member Functions

- **InternalCommandListener** ()
- virtual **~InternalCommandListener** ()
- void **setTransport** (**MockTransport** *transport)
- void **setResponseBuilder** (const **Pointer**< **ResponseBuilder** > &responseBuilder)
- virtual void **onCommand** (const **Pointer**< **Command** > &command)

Event handler for the receipt of a command.

- void **run** ()

6.346.1 Detailed Description

Listens for Commands sent from the **MockTransport** (p. 2227). This class processes all outbound commands and sends responses that are constructed by calling the Protocol provided **ResponseBuilder** (p. 2614) and getting a set of Commands to send back into the **MockTransport** (p. 2227) as incoming Commands and Responses.

6.346.2 Constructor & Destructor Documentation

6.346.2.1 `activemq::transport::mock::InternalCommandListener::InternalCommandListener ()`

6.346.2.2 `virtual
activemq::transport::mock::InternalCommandListener::~~InternalCommandListener
() [virtual]`

6.346.3 Member Function Documentation

6.346.3.1 `virtual void ac-
tivemq::transport::mock::InternalCommandListener::onCommand (
const Pointer< Command > & command) [virtual]`

Event handler for the receipt of a command.

The transport passes off all received commands to its listeners, the listener then owns the Object. If there is no registered listener the **Transport** (p. 3066) deletes the command upon receipt.

Parameters

command the received command object.

Implements **activemq::transport::TransportListener** (p. 3081).

6.346.3.2 void activemq::transport::mock::InternalCommandListener::run ()

6.346.3.3 void activemq::transport::mock::InternalCommandListener::setResponseBuilder (const Pointer< ResponseBuilder > & *responseBuilder*) [inline]

6.346.3.4 void activemq::transport::mock::InternalCommandListener::setTransport (MockTransport * *transport*) [inline]

The documentation for this class was generated from the following file:

- src/main/activemq/transport/mock/**InternalCommandListener.h**

6.347 decaf::lang::exceptions::InterruptedException Class Reference

```
#include <src/main/decaf/lang/exceptions/InterruptedException.h>
```

Inheritance diagram for decaf::lang::exceptions::InterruptedException:

Public Member Functions

- **InterruptedException** () throw ()
Default Constructor.
- **InterruptedException** (const **Exception** &ex) throw ()
*Conversion Constructor from some other **Exception** (p. 1476).*
- **InterruptedException** (const **InterruptedException** &ex) throw ()
Copy Constructor.
- **InterruptedException** (const char *file, const int lineNumber, const std::exception ***cause**, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **InterruptedException** (const std::exception ***cause**) throw ()
Constructor.
- **InterruptedException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **InterruptedException** * **clone** () const
Clones this exception.
- virtual ~**InterruptedException** () throw ()

6.347.1 Constructor & Destructor Documentation

6.347.1.1 decaf::lang::exceptions::InterruptedException::InterruptedException () throw () [inline]

Default Constructor.

6.347.1.2 decaf::lang::exceptions::InterruptedException::InterruptedException (const Exception & *ex*) throw () [inline]

Conversion Constructor from some other **Exception** (p. 1476).

Parameters

ex The **Exception** (p. 1476) whose data is to be copied into this one.

6.347.1.3 decaf::lang::exceptions::InterruptedException::InterruptedException (const InterruptedException & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex The **Exception** (p. 1476) whose data is to be copied into this one.

6.347.1.4 decaf::lang::exceptions::InterruptedException::InterruptedException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.347.1.5 decaf::lang::exceptions::InterruptedException::InterruptedException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters

cause **Pointer** (p. 2343) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.347.1.6 `decaf::lang::exceptions::InterruptedException::InterruptedException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.347.1.7 `virtual decaf::lang::exceptions::InterruptedException::~~InterruptedException () throw () [inline, virtual]`

6.347.2 Member Function Documentation

6.347.2.1 `virtual InterruptedException* decaf::lang::exceptions::InterruptedException::clone () const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

an new **Exception** (p. 1476) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1479).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/exceptions/InterruptedException.h`

6.348 decaf::io::InterruptedException Class Reference

```
#include <src/main/decaf/io/InterruptedException.h>
```

Inheritance diagram for `decaf::io::InterruptedException`:

Public Member Functions

- **InterruptedException** () throw ()
Default Constructor.

- **InterruptedException** (const lang::Exception &ex) throw ()
Copy Constructor.
- **InterruptedException** (const InterruptedException &ex) throw ()
Copy Constructor.
- **InterruptedException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **InterruptedException** (const std::exception *cause) throw ()
Constructor.
- **InterruptedException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor.
- virtual **InterruptedException * clone** () const
Clones this exception.
- virtual **~InterruptedException** () throw ()

6.348.1 Constructor & Destructor Documentation

6.348.1.1 decaf::io::InterruptedException::InterruptedException () throw () [inline]

Default Constructor.

6.348.1.2 decaf::io::InterruptedException::InterruptedException (const lang::Exception & ex) throw () [inline]

Copy Constructor.

Parameters

ex the exception to copy

6.348.1.3 decaf::io::InterruptedException::InterruptedException (const InterruptedException & ex) throw () [inline]

Copy Constructor.

Parameters

ex the exception to copy, which is an instance of this type

6.348.1.4 `decaf::io::InterruptedIOException::InterruptedIOException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.348.1.5 `decaf::io::InterruptedIOException::InterruptedIOException (const std::exception * cause) throw () [inline]`

Constructor.

Parameters

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.348.1.6 `decaf::io::InterruptedIOException::InterruptedIOException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor.

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.348.1.7 `virtual decaf::io::InterruptedIOException::~~InterruptedIOException () throw () [inline, virtual]`

6.348.2 Member Function Documentation

6.348.2.1 `virtual InterruptedIOException* decaf::io::InterruptedIOException::clone () const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

a new exception that is a copy of this one.

Reimplemented from **decaf::io::IOException** (p. 1709).

Reimplemented in **decaf::net::SocketTimeoutException** (p. 2811).

The documentation for this class was generated from the following file:

- src/main/decaf/io/**InterruptedIOException.h**

6.349 cms::InvalidClientIdException Class Reference

This exception must be thrown when a client attempts to set a connection's client ID to a value that is rejected by a provider.

```
#include <src/main/cms/InvalidClientIdException.h>
```

Inheritance diagram for cms::InvalidClientIdException:

Public Member Functions

- **InvalidClientIdException** () throw ()
- **InvalidClientIdException** (const **InvalidClientIdException** &ex) throw ()
- **InvalidClientIdException** (const std::string &message, const std::exception *cause) throw ()
- **InvalidClientIdException** (const std::string &message, const std::exception *cause, const std::vector< std::pair< std::string, int > > &stackTrace) throw ()
- virtual ~**InvalidClientIdException** () throw ()

6.349.1 Detailed Description

This exception must be thrown when a client attempts to set a connection's client ID to a value that is rejected by a provider.

Since

1.3

6.349.2 Constructor & Destructor Documentation

- 6.349.2.1 `cms::InvalidClientIdException::InvalidClientIdException () throw ()`
- 6.349.2.2 `cms::InvalidClientIdException::InvalidClientIdException (const InvalidClientIdException & ex) throw ()`
- 6.349.2.3 `cms::InvalidClientIdException::InvalidClientIdException (const std::string & message, const std::exception * cause) throw ()`
- 6.349.2.4 `cms::InvalidClientIdException::InvalidClientIdException (const std::string & message, const std::exception * cause, const std::vector< std::pair< std::string, int > > & stackTrace) throw ()`
- 6.349.2.5 `virtual cms::InvalidClientIdException::~InvalidClientIdException () throw () [virtual]`

The documentation for this class was generated from the following file:

- `src/main/cms/InvalidClientIdException.h`

6.350 cms::InvalidDestinationException Class Reference

This exception must be thrown when a destination either is not understood by a provider or is no longer valid.

```
#include <src/main/cms/InvalidDestinationException.h>
```

Inheritance diagram for cms::InvalidDestinationException:

Public Member Functions

- `InvalidDestinationException () throw ()`
- `InvalidDestinationException (const InvalidDestinationException &ex) throw ()`
- `InvalidDestinationException (const std::string &message, const std::exception *cause) throw ()`
- `InvalidDestinationException (const std::string &message, const std::exception *cause, const std::vector< std::pair< std::string, int > > &stackTrace) throw ()`
- `virtual ~InvalidDestinationException () throw ()`

6.350.1 Detailed Description

This exception must be thrown when a destination either is not understood by a provider or is no longer valid.

Since

1.3

6.350.2 Constructor & Destructor Documentation

- 6.350.2.1 `cms::InvalidDestinationException::InvalidDestinationException ()
throw ()`
- 6.350.2.2 `cms::InvalidDestinationException::InvalidDestinationException (const
InvalidDestinationException & ex) throw ()`
- 6.350.2.3 `cms::InvalidDestinationException::InvalidDestinationException (const
std::string & message, const std::exception * cause) throw ()`
- 6.350.2.4 `cms::InvalidDestinationException::InvalidDestinationException (const
std::string & message, const std::exception * cause, const std::vector<
std::pair< std::string, int > > & stackTrace) throw ()`
- 6.350.2.5 `virtual cms::InvalidDestinationException::~~InvalidDestinationException
() throw () [virtual]`

The documentation for this class was generated from the following file:

- `src/main/cms/InvalidDestinationException.h`

6.351 decaf::security::InvalidKeyException Class Reference

```
#include <src/main/decaf/security/InvalidKeyException.h>
```

Inheritance diagram for decaf::security::InvalidKeyException:

Public Member Functions

- **InvalidKeyException** () throw ()
Default Constructor.
- **InvalidKeyException** (const Exception &ex) throw ()
Conversion Constructor from some other Exception.
- **InvalidKeyException** (const **InvalidKeyException** &ex) throw ()
Copy Constructor.
- **InvalidKeyException** (const char *file, const int lineNumber, const std::exception *cause,
const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **InvalidKeyException** (const std::exception *cause) throw ()
Constructor.
- **InvalidKeyException** (const char *file, const int lineNumber, const char *msg,...) throw
()
Constructor - Initializes the file name and line number where this message occurred.

- virtual **InvalidKeyException** * **clone** () const
Clones this exception.
- virtual ~**InvalidKeyException** () throw ()

6.351.1 Constructor & Destructor Documentation

6.351.1.1 decaf::security::InvalidKeyException::InvalidKeyException () throw () [inline]

Default Constructor.

6.351.1.2 decaf::security::InvalidKeyException::InvalidKeyException (const Exception & *ex*) throw () [inline]

Conversion Constructor from some other Exception.

Parameters

ex An exception that should become this type of Exception

6.351.1.3 decaf::security::InvalidKeyException::InvalidKeyException (const InvalidKeyException & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex An exception that should become this type of Exception

6.351.1.4 decaf::security::InvalidKeyException::InvalidKeyException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.351.1.5 `decaf::security::InvalidKeyException::InvalidKeyException (const std::exception * cause) throw () [inline]`

Constructor.

Parameters

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.351.1.6 `decaf::security::InvalidKeyException::InvalidKeyException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file name where exception occurs

lineNumber line number where the exception occurred.

msg message to report

... list of primitives that are formatted into the message

6.351.1.7 `virtual decaf::security::InvalidKeyException::~~InvalidKeyException () throw () [inline, virtual]`

6.351.2 Member Function Documentation

6.351.2.1 `virtual InvalidKeyException* decaf::security::InvalidKeyException::clone () const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

A deep copy of this exception.

Reimplemented from `decaf::security::KeyException` (p. 1839).

The documentation for this class was generated from the following file:

- `src/main/decaf/security/InvalidKeyException.h`

6.352 decaf::nio::InvalidMarkException Class Reference

```
#include <src/main/decaf/nio/InvalidMarkException.h>
```

Inheritance diagram for `decaf::nio::InvalidMarkException`:

Public Member Functions

- **InvalidMarkException** () throw ()
Default Constructor.
- **InvalidMarkException** (const **lang::Exception** &ex) throw ()
Conversion Constructor from some other Exception.
- **InvalidMarkException** (const **InvalidMarkException** &ex) throw ()
Copy Constructor.
- **InvalidMarkException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **InvalidMarkException** (const std::exception *cause) throw ()
Constructor.
- **InvalidMarkException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **InvalidMarkException** * clone () const
Clones this exception.
- virtual ~**InvalidMarkException** () throw ()

6.352.1 Constructor & Destructor Documentation

6.352.1.1 `decaf::nio::InvalidMarkException::InvalidMarkException () throw ()` [inline]

Default Constructor.

6.352.1.2 `decaf::nio::InvalidMarkException::InvalidMarkException (const lang::Exception & ex) throw ()` [inline]

Conversion Constructor from some other Exception.

Parameters

ex The Exception whose state data is to be copied into this Exception.

6.352.1.3 decaf::nio::InvalidMarkException::InvalidMarkException (const InvalidMarkException & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex The Exception whose state data is to be copied into this Exception.

6.352.1.4 decaf::nio::InvalidMarkException::InvalidMarkException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.352.1.5 decaf::nio::InvalidMarkException::InvalidMarkException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.352.1.6 decaf::nio::InvalidMarkException::InvalidMarkException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.352.1.7 `virtual decaf::nio::InvalidMarkException::~InvalidMarkException ()
throw () [inline, virtual]`

6.352.2 Member Function Documentation

6.352.2.1 `virtual InvalidMarkException* decaf::nio::InvalidMarkException::clone () const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

A new Exception instance that is a copy of this Exception.

Reimplemented from `decaf::lang::exceptions::IllegalStateException` (p. 1620).

The documentation for this class was generated from the following file:

- `src/main/decaf/nio/InvalidMarkException.h`

6.353 cms::InvalidSelectorException Class Reference

This exception must be thrown when a CMS client attempts to give a provider a message selector with invalid syntax.

```
#include <src/main/cms/InvalidSelectorException.h>
```

Inheritance diagram for `cms::InvalidSelectorException`:

Public Member Functions

- `InvalidSelectorException () throw ()`
- `InvalidSelectorException (const InvalidSelectorException &ex) throw ()`
- `InvalidSelectorException (const std::string &message, const std::exception *cause) throw ()`
- `InvalidSelectorException (const std::string &message, const std::exception *cause, const std::vector< std::pair< std::string, int > > &stackTrace) throw ()`
- `virtual ~InvalidSelectorException () throw ()`

6.353.1 Detailed Description

This exception must be thrown when a CMS client attempts to give a provider a message selector with invalid syntax.

Since

1.3

6.353.2 Constructor & Destructor Documentation

- 6.353.2.1 `cms::InvalidSelectorException::InvalidSelectorException () throw ()`
- 6.353.2.2 `cms::InvalidSelectorException::InvalidSelectorException (const InvalidSelectorException & ex) throw ()`
- 6.353.2.3 `cms::InvalidSelectorException::InvalidSelectorException (const std::string & message, const std::exception * cause) throw ()`
- 6.353.2.4 `cms::InvalidSelectorException::InvalidSelectorException (const std::string & message, const std::exception * cause, const std::vector< std::pair< std::string, int > > & stackTrace) throw ()`
- 6.353.2.5 `virtual cms::InvalidSelectorException::~InvalidSelectorException () throw () [virtual]`

The documentation for this class was generated from the following file:

- `src/main/cms/InvalidSelectorException.h`

6.354 decaf::lang::exceptions::InvalidStateException Class Reference

```
#include <src/main/decaf/lang/exceptions/InvalidStateException.h>
```

Inheritance diagram for decaf::lang::exceptions::InvalidStateException:

Public Member Functions

- **InvalidStateException** () throw ()
Default Constructor.
- **InvalidStateException** (const **Exception** &ex) throw ()
*Conversion Constructor from some other **Exception** (p. 1476).*
- **InvalidStateException** (const **InvalidStateException** &ex) throw ()
Copy Constructor.
- **InvalidStateException** (const char *file, const int lineNumber, const std::exception ***cause**, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **InvalidStateException** (const std::exception ***cause**) throw ()
Constructor.
- **InvalidStateException** (const char *file, const int lineNumber, const char *msg,...) throw ()

Constructor - Initializes the file name and line number where this message occurred.

- virtual **InvalidStateException** * **clone** () const
Clones this exception.
- virtual ~**InvalidStateException** () throw ()

6.354.1 Constructor & Destructor Documentation

6.354.1.1 decaf::lang::exceptions::InvalidStateException::InvalidStateException () throw () [inline]

Default Constructor.

6.354.1.2 decaf::lang::exceptions::InvalidStateException::InvalidStateException (const Exception & *ex*) throw () [inline]

Conversion Constructor from some other **Exception** (p. 1476).

Parameters

ex The **Exception** (p. 1476) whose data is to be copied into this one.

6.354.1.3 decaf::lang::exceptions::InvalidStateException::InvalidStateException (const InvalidStateException & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex The **Exception** (p. 1476) whose data is to be copied into this one.

6.354.1.4 decaf::lang::exceptions::InvalidStateException::InvalidStateException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.354.1.5 decaf::lang::exceptions::InvalidStateException::InvalidStateException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters

***cause* Pointer** (p. 2343) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.354.1.6 decaf::lang::exceptions::InvalidStateException::InvalidStateException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.354.1.7 virtual decaf::lang::exceptions::InvalidStateException::~~InvalidStateException () throw () [inline, virtual]

6.354.2 Member Function Documentation

6.354.2.1 virtual InvalidStateException* decaf::lang::exceptions::InvalidStateException::clone () const [inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

an new **Exception** (p. 1476) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1479).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/exceptions/**InvalidStateException.h**

6.355 decaf::io::IOException Class Reference

```
#include <src/main/decaf/io/IOException.h>
```

Inheritance diagram for decaf::io::IOException:

Public Member Functions

- **IOException** () throw ()
Default Constructor.
- **IOException** (const lang::Exception &ex) throw ()
Copy Constructor.
- **IOException** (const IOException &ex) throw ()
Copy Constructor.
- **IOException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **IOException** (const std::exception *cause) throw ()
Constructor.
- **IOException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor.
- virtual **IOException** * clone () const
Clones this exception.
- virtual ~**IOException** () throw ()

6.355.1 Constructor & Destructor Documentation

6.355.1.1 decaf::io::IOException::IOException () throw () [inline]

Default Constructor.

6.355.1.2 decaf::io::IOException::IOException (const lang::Exception & ex) throw () [inline]

Copy Constructor.

Parameters

ex the exception to copy

6.355.1.3 `decaf::io::IOException::IOException (const IOException & ex) throw () [inline]`

Copy Constructor.

Parameters

ex the exception to copy, which is an instance of this type

6.355.1.4 `decaf::io::IOException::IOException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.355.1.5 `decaf::io::IOException::IOException (const std::exception * cause) throw () [inline]`

Constructor.

Parameters

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.355.1.6 `decaf::io::IOException::IOException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor.

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.355.1.7 `virtual decaf::io::IOException::~~IOException () throw () [inline, virtual]`

6.355.2 Member Function Documentation

6.355.2.1 `virtual IOException* decaf::io::IOException::clone () const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

a new instance of an Exception that is a copy of this instance.

Reimplemented from `decaf::lang::Exception` (p. 1479).

Reimplemented in `decaf::io::EOFException` (p. 1476), `decaf::io::InterruptedIOException` (p. 1695), `decaf::io::UnsupportedEncodingException` (p. 3092), `decaf::io::UTFDataFormatException` (p. 3141), `decaf::net::BindException` (p. 665), `decaf::net::ConnectException` (p. 1051), `decaf::net::HttpRetryException` (p. 1613), `decaf::net::MalformedURLException` (p. 1970), `decaf::net::NoRouteToHostException` (p. 2271), `decaf::net::PortUnreachableException` (p. 2370), `decaf::net::ProtocolException` (p. 2506), `decaf::net::SocketException` (p. 2794), `decaf::net::SocketTimeoutException` (p. 2811), `decaf::net::UnknownHostException` (p. 3087), and `decaf::net::UnknownServiceException` (p. 3089).

The documentation for this class was generated from the following file:

- `src/main/decaf/io/IOException.h`

6.356 activemq::transport::IOTransport Class Reference

Implementation of the **Transport** (p. 3066) interface that performs marshaling of commands to IO streams.

```
#include <src/main/activemq/transport/IOTransport.h>
```

Inheritance diagram for `activemq::transport::IOTransport`:

Public Member Functions

- **IOTransport** ()
Default Constructor.
- **IOTransport** (const **Pointer**< **wireformat::WireFormat** > &wireFormat)
*Create an instance of this **Transport** (p. 3066) and assign its **WireFormat** instance at creation time.*
- virtual **~IOTransport** ()

- virtual void **oneway** (const **Pointer**< **Command** > &command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)
Sends a one-way command.
- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)
Not supported by this class - throws an exception.
- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command, unsigned int timeout) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)
Not supported by this class - throws an exception.
- virtual void **setWireFormat** (const **Pointer**< **wireformat::WireFormat** > &wireFormat)
Sets the WireFormat instance to use.
- virtual void **setTransportListener** (**TransportListener** *listener)
Sets the observer of asynchronous exceptions from this transport.
- virtual **TransportListener** * **getTransportListener** () const
Gets the observer of asynchronous exceptions from this transport.
- virtual void **setInputStream** (decaf::io::DataInputStream *is)
Sets the input stream for in-coming commands.
- virtual void **setOutputStream** (decaf::io::DataOutputStream *os)
Sets the output stream for out-going commands.
- virtual void **start** () throw (decaf::io::IOException)
Starts this transport object and creates the thread for polling on the input stream for commands.
- virtual void **stop** () throw (decaf::io::IOException)
*Stops the **Transport** (p. 3066), terminating any threads and stopping all read and write operations.*
- virtual void **close** () throw (decaf::io::IOException)
Stops the polling thread and closes the streams.
- virtual void **run** ()
Runs the polling thread.
- virtual **Transport** * **narrow** (const std::type_info &typeId)
*Narrows down a Chain of Transports to a specific **Transport** (p. 3066) to allow a higher level transport to skip intermediate Transports in certain circumstances.*
- virtual bool **isFaultTolerant** () const
*Is this **Transport** (p. 3066) fault tolerant, meaning that it will reconnect to a broker on disconnect.*
- virtual bool **isConnected** () const

Is the **Transport** (p. 3066) Connected to its Broker.

- virtual bool **isClosed** () const

Has the **Transport** (p. 3066) been shutdown and no longer usable.

- virtual std::string **getRemoteAddress** () const
- virtual void **reconnect** (const **decaf::net::URI** &uri AMQCPP_UNUSED) throw (**decaf::io::IOException**)
reconnect to another location

6.356.1 Detailed Description

Implementation of the **Transport** (p. 3066) interface that performs marshaling of commands to IO streams. This class does not implement the request method, it only handles oneway messages. A thread polls on the input stream for in-coming commands. When a command is received, the command listener is notified. The polling thread is not started until the start method is called. The close method will close the associated streams. Close can be called explicitly by the user, but is also called in the destructor. Once this object has been closed, it cannot be restarted.

6.356.2 Constructor & Destructor Documentation

6.356.2.1 **activemq::transport::IOTransport::IOTransport** ()

Default Constructor.

6.356.2.2 **activemq::transport::IOTransport::IOTransport** (const **Pointer**< **wireformat::WireFormat** > & *wireFormat*)

Create an instance of this **Transport** (p. 3066) and assign its **WireFormat** instance at creation time.

Parameters

wireFormat Data encoder / decoder to use when reading and writing.

6.356.2.3 **virtual activemq::transport::IOTransport::~~IOTransport** () [virtual]

6.356.3 Member Function Documentation

6.356.3.1 **virtual void activemq::transport::IOTransport::close** () throw (**decaf::io::IOException**) [virtual]

Stops the polling thread and closes the streams.

This can be called explicitly, but is also called in the destructor. Once this object has been closed, it cannot be restarted.

Exceptions

IOException if errors occur.

6.356.3.2 `virtual std::string activemq::transport::IOTransport::getRemoteAddress () const [inline, virtual]`

Returns

the remote address for this connection

6.356.3.3 `virtual TransportListener* activemq::transport::IOTransport::getTransportListener () const [inline, virtual]`

Gets the observer of asynchronous exceptions from this transport.

Returns

The listener of transport events.

6.356.3.4 `virtual bool activemq::transport::IOTransport::isClosed () const [inline, virtual]`

Has the **Transport** (p. 3066) been shutdown and no longer usable.

Returns

true if the **Transport** (p. 3066)

6.356.3.5 `virtual bool activemq::transport::IOTransport::isConnected () const [inline, virtual]`

Is the **Transport** (p. 3066) Connected to its Broker.

Returns

true if a connection has been made.

6.356.3.6 `virtual bool activemq::transport::IOTransport::isFaultTolerant () const [inline, virtual]`

Is this **Transport** (p. 3066) fault tolerant, meaning that it will reconnect to a broker on disconnect.

Returns

true if the **Transport** (p. 3066) is fault tolerant.

6.356.3.7 `virtual Transport* activemq::transport::IOTransport::narrow (const std::type_info & typeId) [inline, virtual]`

Narrows down a Chain of Transports to a specific **Transport** (p. 3066) to allow a higher level transport to skip intermediate Transports in certain circumstances.

Parameters

typeId - The type_info of the Object we are searching for.

Returns

the requested Object. or NULL if its not in this chain.

6.356.3.8 `virtual void activemq::transport::IOTransport::oneway (const
Pointer< Command > & command) throw (decaf::io::IOException,
decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Sends a one-way command.

Does not wait for any response from the broker.

Parameters

command the command to be sent.

Exceptions

IOException if an exception occurs during writing of the command.

UnsupportedOperationException if this method is not implemented by this transport.

6.356.3.9 `virtual void activemq::transport::IOTransport::reconnect (const decaf::net::URI &uri AMQCPP_UNUSED) throw (decaf::io::IOException) [inline, virtual]`

reconnect to another location

Parameters

uri

Exceptions

IOException on failure of if not supported

6.356.3.10 `virtual Pointer<Response> activemq::transport::IOTransport::request (const Pointer< Command > & command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Not supported by this class - throws an exception.

Parameters

command the command to be sent.

Returns

the response to the command sent.

Exceptions

UnsupportedOperationException.

6.356.3.11 `virtual Pointer<Response> activemq::transport::IOTransport::request
 (const Pointer< Command > & command, unsigned
 int timeout) throw (decaf::io::IOException,
 decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Not supported by this class - throws an exception.

Parameters

command the command to be sent.

timeout the time to wait for a response.

Returns

the response to the command sent.

Exceptions

UnsupportedOperationException.

6.356.3.12 `virtual void activemq::transport::IOTransport::run () [virtual]`

Runs the polling thread.

Implements **decaf::lang::Runnable** (p. 2642).

6.356.3.13 `virtual void activemq::transport::IOTransport::setInputStream (
 decaf::io::DataInputStream * is) [inline, virtual]`

Sets the input stream for in-coming commands.

Parameters

is The input stream.

6.356.3.14 `virtual void activemq::transport::IOTransport::setOutputStream (
 decaf::io::DataOutputStream * os) [inline, virtual]`

Sets the output stream for out-going commands.

Parameters

os The output stream.

6.356.3.15 `virtual void activemq::transport::IOTransport::setTransportListener (
 TransportListener * listener) [inline, virtual]`

Sets the observer of asynchronous exceptions from this transport.

Parameters

listener the listener of transport events.

6.356.3.16 `virtual void activemq::transport::IOTransport::setWireFormat (const Pointer< wireformat::WireFormat > & wireFormat) [inline, virtual]`

Sets the WireFormat instance to use.

Parameters

wireFormat The WireFormat the object used to encode / decode commands.

6.356.3.17 `virtual void activemq::transport::IOTransport::start () throw (decaf::io::IOException) [virtual]`

Starts this transport object and creates the thread for polling on the input stream for commands.

If this object has been closed, throws an exception. Before calling start, the caller must set the IO streams and the reader and writer objects.

Exceptions

CMSEException if an error occurs or if this transport has already been closed.

6.356.3.18 `virtual void activemq::transport::IOTransport::stop () throw (decaf::io::IOException) [virtual]`

Stops the **Transport** (p. 3066), terminating any threads and stopping all read and write operations.

Exceptions

IOException if an error occurs while stopping the **Transport** (p. 3066).

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/IOTransport.h`

6.357 decaf::lang::Iterable< E > Class Template Reference

Implementing this interface allows an object to be cast to an **Iterable** (p. 1715) type for generic collections API calls.

```
#include <src/main/decaf/lang/Iterable.h>
```

Inheritance diagram for `decaf::lang::Iterable< E >`:

Public Member Functions

- `virtual ~Iterable ()`
- `virtual decaf::util::Iterator< E > * iterator ()=0`
- `virtual decaf::util::Iterator< E > * iterator () const =0`

6.357.1 Detailed Description

template<typename E> class decaf::lang::Iterable< E >

Implementing this interface allows an object to be cast to an **Iterable** (p. 1715) type for generic collections API calls.

6.357.2 Constructor & Destructor Documentation

6.357.2.1 **template<typename E> virtual decaf::lang::Iterable< E >::~~Iterable ()** [inline, virtual]

6.357.3 Member Function Documentation

6.357.3.1 **template<typename E> virtual decaf::util::Iterator<E>***
decaf::lang::Iterable< E >::iterator () [pure virtual]

Returns

an iterator over a set of elements of type T.

Referenced by decaf::util::AbstractCollection< Pointer< BackupTransport > >::clear(), decaf::util::AbstractCollection< Pointer< BackupTransport > >::contains(), decaf::util::AbstractCollection< Pointer< BackupTransport > >::copy(), decaf::util::AbstractCollection< Pointer< BackupTransport > >::operator=(), decaf::util::AbstractCollection< Pointer< BackupTransport > >::remove(), decaf::util::AbstractSet< ActiveMQSession * >::removeAll(), decaf::util::AbstractCollection< Pointer< BackupTransport > >::removeAll(), decaf::util::AbstractCollection< Pointer< BackupTransport > >::retainAll(), and decaf::util::AbstractCollection< Pointer< BackupTransport > >::toArray().

6.357.3.2 **template<typename E> virtual decaf::util::Iterator<E>***
decaf::lang::Iterable< E >::iterator () const [pure virtual]

The documentation for this class was generated from the following file:

- src/main/decaf/lang/Iterable.h

6.358 decaf::util::Iterator< T > Class Template Reference

Defines an object that can be used to iterate over the elements of a collection.

#include <src/main/decaf/util/Iterator.h>

Inheritance diagram for decaf::util::Iterator< T >:

Public Member Functions

- virtual ~Iterator ()

- virtual T **next** ()=0 throw (lang::exceptions::NoSuchElementException)
Returns the next element in the iteration.
- virtual bool **hasNext** () const =0
Returns true if the iteration has more elements.
- virtual void **remove** ()=0 throw (lang::exceptions::IllegalStateException, lang::exceptions::UnsupportedOperationException)
Removes from the underlying collection the last element returned by the iterator (optional operation).

6.358.1 Detailed Description

template<typename T> class decaf::util::Iterator< T >

Defines an object that can be used to iterate over the elements of a collection. The iterator provides a way to access and remove elements with well defined semantics.

6.358.2 Constructor & Destructor Documentation

6.358.2.1 **template<typename T> virtual decaf::util::Iterator< T >::~~Iterator ()**
[inline, virtual]

6.358.3 Member Function Documentation

6.358.3.1 **template<typename T> virtual bool decaf::util::Iterator< T >::hasNext () const** [pure virtual]

Returns true if the iteration has more elements.

Returns false if the next call to next would result in an NoSuchElementException to be thrown.

6.358.3.2 **template<typename T> virtual T decaf::util::Iterator< T >::next ()**
throw (lang::exceptions::NoSuchElementException) [pure virtual]

Returns the next element in the iteration.

Calling this method repeatedly until the **hasNext()** (p.1717) method returns false will return each element in the underlying collection exactly once.

Returns

next element in the iteration of elements

Exceptions

NoSuchElementException - iteration has no more elements.

6.358.3.3 `template<typename T> virtual void decaf::util::Iterator< T
>::remove () throw (lang::exceptions::IllegalStateException,
lang::exceptions::UnsupportedOperationException) [pure virtual]`

Removes from the underlying collection the last element returned by the iterator (optional operation).

This method can be called only once per call to next. The behavior of an iterator is unspecified if the underlying collection is modified while the iteration is in progress in any way other than by calling this method.

Exceptions

UnsupportedOperationException - if the remove operation is not supported by this **Iterator** (p. 1716).

IllegalStateException - if the next method has not yet been called, or the remove method has already been called after the last call to the next method.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/Iterator.h`

6.359 activemq::commands::JournalQueueAck Class Reference

```
#include <src/main/activemq/commands/JournalQueueAck.h>
```

Inheritance diagram for `activemq::commands::JournalQueueAck`:

Public Member Functions

- **JournalQueueAck** ()
- virtual **~JournalQueueAck** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **JournalQueueAck * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1372) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent.*

- virtual const **Pointer**< **ActiveMQDestination** > & **getDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &destination)
- virtual const **Pointer**< **MessageAck** > & **getMessageAck** () const
- virtual **Pointer**< **MessageAck** > & **getMessageAck** ()
- virtual void **setMessageAck** (const **Pointer**< **MessageAck** > &messageAck)

Static Public Attributes

- static const unsigned char **ID_JOURNALQUEUEACK** = 52

Protected Member Functions

- **JournalQueueAck** (const **JournalQueueAck** &)
- **JournalQueueAck** & **operator=** (const **JournalQueueAck** &)

Protected Attributes

- **Pointer**< **ActiveMQDestination** > **destination**
- **Pointer**< **MessageAck** > **messageAck**

6.359.1 Constructor & Destructor Documentation

6.359.1.1 **activemq::commands::JournalQueueAck::JournalQueueAck** (const **JournalQueueAck** &) [inline, protected]

6.359.1.2 **activemq::commands::JournalQueueAck::JournalQueueAck** ()

6.359.1.3 **virtual activemq::commands::JournalQueueAck::~~JournalQueueAck** () [virtual]

6.359.2 Member Function Documentation

6.359.2.1 **virtual JournalQueueAck* activemq::commands::JournalQueueAck::cloneDataStructure** () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1373).

6.359.2.2 `virtual void activemq::commands::JournalQueueAck::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

Implements `activemq::commands::DataStructure` (p. 1374).

6.359.2.3 `virtual bool activemq::commands::JournalQueueAck::equals (const DataStructure * value) const [virtual]`

Compares the `DataStructure` (p. 1372) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if `DataStructure`'s are Equal.

Implements `activemq::commands::DataStructure` (p. 1374).

6.359.2.4 `virtual unsigned char activemq::commands::JournalQueueAck::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

Returns

new `DataStructure` (p. 1372) type copy.

Implements `activemq::commands::DataStructure` (p. 1375).

- 6.359.2.5 `virtual Pointer<ActiveMQDestination>& activemq::commands::JournalQueueAck::getDestination ()`
[virtual]
- 6.359.2.6 `virtual const Pointer<ActiveMQDestination>& activemq::commands::JournalQueueAck::getDestination () const`
[virtual]
- 6.359.2.7 `virtual const Pointer<MessageAck>& activemq::commands::JournalQueueAck::getMessageAck () const` [virtual]
- 6.359.2.8 `virtual Pointer<MessageAck>& activemq::commands::JournalQueueAck::getMessageAck ()` [virtual]
- 6.359.2.9 `JournalQueueAck& activemq::commands::JournalQueueAck::operator= (const JournalQueueAck &)` [inline, protected]
- 6.359.2.10 `virtual void activemq::commands::JournalQueueAck::setDestination (const Pointer< ActiveMQDestination > & destination)` [virtual]
- 6.359.2.11 `virtual void activemq::commands::JournalQueueAck::setMessageAck (const Pointer< MessageAck > & messageAck)` [virtual]
- 6.359.2.12 `virtual std::string activemq::commands::JournalQueueAck::toString () const` [virtual]

Returns a string containing the information for this **DataStructure** (p.1372) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from `activemq::commands::BaseDataStructure` (p.662).

6.359.3 Field Documentation

- 6.359.3.1 `Pointer<ActiveMQDestination> activemq::commands::JournalQueueAck::destination`
[protected]
- 6.359.3.2 `const unsigned char activemq::commands::JournalQueueAck::ID _ - JOURNALQUEUEACK = 52` [static]
- 6.359.3.3 `Pointer<MessageAck> activemq::commands::JournalQueueAck::messageAck`
[protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/JournalQueueAck.h`

6.360 activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller Class Reference

Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p.1722).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/JournalQueueAckMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller:

Public Member Functions

- **JournalQueueAckMarshaller** ()
- virtual **~JournalQueueAckMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.360.1 Detailed Description

Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 1722). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.360.2 Constructor & Destructor Documentation

6.360.2.1 `activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller::JournalQueueAckMarshaller () [inline]`

6.360.2.2 `virtual activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller::~~JournalQueueAckMarshaller () [inline, virtual]`

6.360.3 Member Function Documentation

6.360.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.360.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.360.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1342).

6.360.3.4 virtual void **activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller::looseUnmarshal**
(**OpenWireFormat** * *wireFormat*, **commands::DataStructure** *
dataStructure, **decaf::io::DataInputStream** * *dataIn*) throw (
decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1348).

6.360.3.5 virtual int **activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller::tightMarshal1**
(**OpenWireFormat** * *wireFormat*, **commands::DataStructure**
* *dataStructure*, **utils::BooleanStream** * *bs*) throw (
decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1354).

6.360.3.6 virtual void activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1360).

6.360.3.7 virtual void activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1366).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/JournalQueueAckMarshaller.h`

6.361 activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller Class Reference

Marshaling code for Open Wire Format for `JournalQueueAckMarshaller` (p. 1725).

#include <src/main/activemq/wireformat/openwire/marshal/v4/JournalQueueAckMarshaller.h>

Inheritance diagram for activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller:

Public Member Functions

- **JournalQueueAckMarshaller** ()
- virtual **~JournalQueueAckMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.361.1 Detailed Description

Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p.1725). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.361.2 Constructor & Destructor Documentation

6.361.2.1 `activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller::JournalQueueAckMarshaller () [inline]`

6.361.2.2 `virtual activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller::~~JournalQueueAckMarshaller () [inline, virtual]`

6.361.3 Member Function Documentation

6.361.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.361.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.361.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1342).

6.361.3.4 virtual void activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1348).

6.361.3.5 virtual int activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1354).

6.361.3.6 virtual void activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1360).

```
6.361.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1366).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/JournalQueueAckMarshaller.h`

6.362 **activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller** Class Reference

Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 1729).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/JournalQueueAckMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller**:

Public Member Functions

- **JournalQueueAckMarshaller** ()
- virtual **~JournalQueueAckMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.362.1 Detailed Description

Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p.1729). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.362.2 Constructor & Destructor Documentation

6.362.2.1 `activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller::JournalQueueAckMarshaller () [inline]`

6.362.2.2 `virtual activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller::~~JournalQueueAckMarshaller () [inline, virtual]`

6.362.3 Member Function Documentation

6.362.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.362.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.362.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1342).

6.362.3.4 virtual void activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1348).

6.362.3.5 virtual int activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1354).

6.362.3.6 virtual void activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1360).

```
6.362.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1366).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/JournalQueueAckMarshaller.h`

6.363 **activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller** Class Reference

Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 1733).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/JournalQueueAckMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller**:

Public Member Functions

- **JournalQueueAckMarshaller** ()
- virtual **~JournalQueueAckMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.363.1 Detailed Description

Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p.1733). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.363.2 Constructor & Destructor Documentation

6.363.2.1 `activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller::JournalQueueAckMarshaller () [inline]`

6.363.2.2 `virtual activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller::~~JournalQueueAckMarshaller () [inline, virtual]`

6.363.3 Member Function Documentation

6.363.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.363.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.363.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1342).

6.363.3.4 virtual void activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1348).

6.363.3.5 virtual int activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1354).

6.363.3.6 virtual void activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1360).

```
6.363.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1366).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/JournalQueueAckMarshaller.h`

6.364 **activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller** Class Reference

Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 1737).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/JournalQueueAckMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller**:

Public Member Functions

- **JournalQueueAckMarshaller** ()
- virtual **~JournalQueueAckMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.364.1 Detailed Description

Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p.1737). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.364.2 Constructor & Destructor Documentation

6.364.2.1 `activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller::JournalQueueAckMarshaller () [inline]`

6.364.2.2 `virtual activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller::~~JournalQueueAckMarshaller () [inline, virtual]`

6.364.3 Member Function Documentation

6.364.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.364.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.364.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1342).

6.364.3.4 virtual void activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1348).

6.364.3.5 virtual int activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1354).

6.364.3.6 virtual void activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1366).

6.364.3.7 `virtual void activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1366).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/JournalQueueAckMarshaller.h`

6.365 `activemq::commands::JournalTopicAck` Class Reference

```
#include <src/main/activemq/commands/JournalTopicAck.h>
```

Inheritance diagram for `activemq::commands::JournalTopicAck`:

Public Member Functions

- `JournalTopicAck ()`
- `virtual ~JournalTopicAck ()`

- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **JournalTopicAck** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1372) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **ActiveMQDestination** > & **getDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &destination)
- virtual const **Pointer**< **MessageId** > & **getMessageId** () const
- virtual **Pointer**< **MessageId** > & **getMessageId** ()
- virtual void **setMessageId** (const **Pointer**< **MessageId** > &messageId)
- virtual long long **getMessageSequenceId** () const
- virtual void **setMessageSequenceId** (long long messageSequenceId)
- virtual const std::string & **getSubscriptionName** () const
- virtual std::string & **getSubscriptionName** ()
- virtual void **setSubscriptionName** (const std::string &subscriptionName)
- virtual const std::string & **getClientId** () const
- virtual std::string & **getClientId** ()
- virtual void **setClientId** (const std::string &clientId)
- virtual const **Pointer**< **TransactionId** > & **getTransactionId** () const
- virtual **Pointer**< **TransactionId** > & **getTransactionId** ()
- virtual void **setTransactionId** (const **Pointer**< **TransactionId** > &transactionId)

Static Public Attributes

- static const unsigned char **ID_JOURNALTOPICACK** = 50

Protected Member Functions

- **JournalTopicAck** (const **JournalTopicAck** &)
- **JournalTopicAck** & **operator=** (const **JournalTopicAck** &)

Protected Attributes

- **Pointer**< **ActiveMQDestination** > destination
- **Pointer**< **MessageId** > messageId
- long long messageSequenceId
- std::string subscriptionName
- std::string clientId
- **Pointer**< **TransactionId** > transactionId

6.365.1 Constructor & Destructor Documentation

- 6.365.1.1** `activemq::commands::JournalTopicAck::JournalTopicAck (const JournalTopicAck &) [inline, protected]`
- 6.365.1.2** `activemq::commands::JournalTopicAck::JournalTopicAck ()`
- 6.365.1.3** `virtual activemq::commands::JournalTopicAck::~~JournalTopicAck () [virtual]`

6.365.2 Member Function Documentation

- 6.365.2.1** `virtual JournalTopicAck* activemq::commands::JournalTopicAck::cloneDataStructure () const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1373).

- 6.365.2.2** `virtual void activemq::commands::JournalTopicAck::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

Implements `activemq::commands::DataStructure` (p. 1374).

- 6.365.2.3** `virtual bool activemq::commands::JournalTopicAck::equals (const DataStructure * value) const [virtual]`

Compares the `DataStructure` (p. 1372) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Implements `activemq::commands::DataStructure` (p. 1374).

6.365.2.4 `virtual const std::string& activemq::commands::JournalTopicAck::getClientId () const [virtual]`

6.365.2.5 `virtual std::string& activemq::commands::JournalTopicAck::getClientId () [virtual]`

6.365.2.6 `virtual unsigned char activemq::commands::JournalTopicAck::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataSetructure** (p. 1372) type copy.

Implements **activemq::commands::DataSetructure** (p. 1375).

- 6.365.2.7 `virtual Pointer<ActiveMQDestination>& activemq::commands::JournalTopicAck::getDestination ()` [virtual]
- 6.365.2.8 `virtual const Pointer<ActiveMQDestination>& activemq::commands::JournalTopicAck::getDestination () const` [virtual]
- 6.365.2.9 `virtual const Pointer<MessageId>& activemq::commands::JournalTopicAck::getMessageId () const` [virtual]
- 6.365.2.10 `virtual Pointer<MessageId>& activemq::commands::JournalTopicAck::getMessageId ()` [virtual]
- 6.365.2.11 `virtual long long activemq::commands::JournalTopicAck::getMessageSequenceId () const` [virtual]
- 6.365.2.12 `virtual std::string& activemq::commands::JournalTopicAck::getSubscriptionName ()` [virtual]
- 6.365.2.13 `virtual const std::string& activemq::commands::JournalTopicAck::getSubscriptionName () const` [virtual]
- 6.365.2.14 `virtual const Pointer<TransactionId>& activemq::commands::JournalTopicAck::getTransactionId () const` [virtual]
- 6.365.2.15 `virtual Pointer<TransactionId>& activemq::commands::JournalTopicAck::getTransactionId ()` [virtual]
- 6.365.2.16 `JournalTopicAck& activemq::commands::JournalTopicAck::operator= (const JournalTopicAck &)` [inline, protected]
- 6.365.2.17 `virtual void activemq::commands::JournalTopicAck::setClientId (const std::string & clientId)` [virtual]
- 6.365.2.18 `virtual void activemq::commands::JournalTopicAck::setDestination (const Pointer< ActiveMQDestination > & destination)` [virtual]
- 6.365.2.19 `virtual void activemq::commands::JournalTopicAck::setMessageId (const Pointer< MessageId > & messageId)` [virtual]
- 6.365.2.20 `virtual void activemq::commands::JournalTopicAck::setMessageSequenceId (long long messageSequenceId)` [virtual]
- 6.365.2.21 `virtual void activemq::commands::JournalTopicAck::setSubscriptionName (const std::string & subscriptionName)` [virtual]
-
- 6.365.2.22 `virtual void activemq::commands::JournalTopicAck::setTransactionId (const Pointer< TransactionId > & transactionId)` [virtual]
- 6.365.2.23 `virtual std::string activemq::commands::JournalTopicAck::toString () const` [virtual]

Returns

formatted string useful for debugging.

Reimplemented from `activemq::commands::BaseDataStructure` (p. 662).

6.365.3 Field Documentation

- 6.365.3.1** `std::string activemq::commands::JournalTopicAck::clientId` [protected]
- 6.365.3.2** `Pointer<ActiveMQDestination> activemq::commands::JournalTopicAck::destination` [protected]
- 6.365.3.3** `const unsigned char activemq::commands::JournalTopicAck::ID_ - JOURNALTOPICACK = 50` [static]
- 6.365.3.4** `Pointer<MessageId> activemq::commands::JournalTopicAck::messageId` [protected]
- 6.365.3.5** `long long activemq::commands::JournalTopicAck::messageSequenceId` [protected]
- 6.365.3.6** `std::string activemq::commands::JournalTopicAck::subscriptionName` [protected]
- 6.365.3.7** `Pointer<TransactionId> activemq::commands::JournalTopicAck::transactionId` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/JournalTopicAck.h`

6.366 activemq::wireformat::openwire::marshal::v2::JournalTopicAckMa Class Reference

Marshaling code for Open Wire Format for `JournalTopicAckMarshaller` (p. 1746).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/JournalTopicAckMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller`:

Public Member Functions

- `JournalTopicAckMarshaller ()`
- `virtual ~JournalTopicAckMarshaller ()`
- `virtual commands::DataStructure * createObject () const`

Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.366.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p.1746). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.366.2 Constructor & Destructor Documentation

6.366.2.1 `activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller::JournalTopicAckMarshaller () [inline]`

6.366.2.2 `virtual
activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller::~~JournalTopicAckMarshaller () [inline, virtual]`

6.366.3 Member Function Documentation

6.366.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.366.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.366.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1342).

6.366.3.4 virtual void activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1348).

6.366.3.5 virtual int activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1354).

6.366.3.6 virtual void activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1360).

6.366.3.7 virtual void activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1366).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/JournalTopicAckMarshaller.h

6.367 activemq::wireformat::openwire::marshal::v1::JournalTopicAckMa Class Reference

Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 1750).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/JournalTopicAckMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller:

Public Member Functions

- **JournalTopicAckMarshaller** ()
- virtual **~JournalTopicAckMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.367.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p.1750). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.367.2 Constructor & Destructor Documentation

6.367.2.1 `activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller::JournalTopicAckMarshaller () [inline]`

6.367.2.2 `virtual
activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller::~~JournalTopicAckMarshaller () [inline, virtual]`

6.367.3 Member Function Documentation

6.367.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.367.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.367.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1342).

6.367.3.4 virtual void activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1348).

6.367.3.5 virtual int activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1354).

6.367.3.6 virtual void activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1360).

6.367.3.7 virtual void ac-
 tivemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller::tightUnmarshal
 (OpenWireFormat * *wireFormat*, commands::DataStructure
 * *dataStructure*, decaf::io::DataInputStream * *dataIn*,
 utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1366).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/JournalTopicAckMarshaller.h

6.368 activemq::wireformat::openwire::marshal::v3::JournalTopicAckMa Class Reference

Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 1754).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/JournalTopicAckMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller:

Public Member Functions

- **JournalTopicAckMarshaller** ()
- virtual **~JournalTopicAckMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.368.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p.1754). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.368.2 Constructor & Destructor Documentation

6.368.2.1 `activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller::JournalTopicAckMarshaller()` [inline]

6.368.2.2 `virtual activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller::~~JournalTopicAckMarshaller()` [inline, virtual]

6.368.3 Member Function Documentation

6.368.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.368.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.368.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1342).

6.368.3.4 virtual void activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1348).

6.368.3.5 virtual int activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1354).

6.368.3.6 virtual void activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1360).

6.368.3.7 virtual void activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1366).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/JournalTopicAckMarshaller.h

6.369 activemq::wireformat::openwire::marshal::v4::JournalTopicAckMa Class Reference

Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p.1758).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/JournalTopicAckMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller:

Public Member Functions

- **JournalTopicAckMarshaller** ()
- virtual **~JournalTopicAckMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.369.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p.1758). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.369.2 Constructor & Destructor Documentation

6.369.2.1 `activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller::JournalTopicAckMarshaller () [inline]`

6.369.2.2 `virtual
activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller::~~JournalTopicAckMarshaller () [inline, virtual]`

6.369.3 Member Function Documentation

6.369.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.369.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.369.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1342).

6.369.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1348).

6.369.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1354).

6.369.3.6 `virtual void activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1360).

```
6.369.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1366).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/JournalTopicAckMarshaller.h

6.370 activemq::wireformat::openwire::marshal::v5::JournalTopicAckMa Class Reference

Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 1762).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/JournalTopicAckMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller**:

Public Member Functions

- **JournalTopicAckMarshaller** ()
- virtual **~JournalTopicAckMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.370.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p.1762). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.370.2 Constructor & Destructor Documentation

6.370.2.1 `activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller::JournalTopicAckMarshaller()` [inline]

6.370.2.2 `virtual activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller::~~JournalTopicAckMarshaller()` [inline, virtual]

6.370.3 Member Function Documentation

6.370.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.370.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.370.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1342).

6.370.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1348).

6.370.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1354).

6.370.3.6 `virtual void activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions

- IOException* if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1360).

```
6.370.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1366).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**JournalTopicAckMarshaller.h**

6.371 activemq::commands::JournalTrace Class Reference

```
#include <src/main/activemq/commands/JournalTrace.h>
```

Inheritance diagram for activemq::commands::JournalTrace:

Public Member Functions

- **JournalTrace** ()
- virtual **~JournalTrace** ()
- virtual unsigned char **getDataStructureType** () const

Get the unique identifier that this object and its own Marshaler share.

- virtual **JournalTrace** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1372) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent.*
- virtual const std::string & **getMessage** () const
- virtual std::string & **getMessage** ()
- virtual void **setMessage** (const std::string &message)

Static Public Attributes

- static const unsigned char **ID_JOURNALTRACE** = 53

Protected Member Functions

- **JournalTrace** (const **JournalTrace** &)
- **JournalTrace** & **operator=** (const **JournalTrace** &)

Protected Attributes

- std::string **message**

6.371.1 Constructor & Destructor Documentation

- 6.371.1.1 **activemq::commands::JournalTrace::JournalTrace** (const **JournalTrace** &) [inline, protected]
- 6.371.1.2 **activemq::commands::JournalTrace::JournalTrace** ()
- 6.371.1.3 **virtual activemq::commands::JournalTrace::~~JournalTrace** () [virtual]

6.371.2 Member Function Documentation

- 6.371.2.1 **virtual JournalTrace* activemq::commands::JournalTrace::cloneDataStructure** () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1373).

6.371.2.2 virtual void activemq::commands::JournalTrace::copyDataStructure (const DataStructure * *src*) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

Implements **activemq::commands::DataStructure** (p. 1374).

6.371.2.3 virtual bool activemq::commands::JournalTrace::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Implements **activemq::commands::DataStructure** (p. 1374).

6.371.2.4 virtual unsigned char activemq::commands::JournalTrace::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataStructure** (p. 1372) type copy.

Implements **activemq::commands::DataStructure** (p. 1375).

- 6.371.2.5 `virtual std::string& activemq::commands::JournalTrace::getMessage ()`
[virtual]
- 6.371.2.6 `virtual const std::string& activemq::commands::JournalTrace::getMessage`
`() const` [virtual]
- 6.371.2.7 `JournalTrace& activemq::commands::JournalTrace::operator= (const`
`JournalTrace &)` [inline, protected]
- 6.371.2.8 `virtual void activemq::commands::JournalTrace::setMessage (const`
`std::string & message)` [virtual]
- 6.371.2.9 `virtual std::string activemq::commands::JournalTrace::toString ()`
`const` [virtual]

Returns a string containing the information for this **DataStructure** (p.1372) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from `activemq::commands::BaseDataStructure` (p.662).

6.371.3 Field Documentation

- 6.371.3.1 `const unsigned char activemq::commands::JournalTrace::ID_ -`
`JOURNALTRACE = 53` [static]
- 6.371.3.2 `std::string activemq::commands::JournalTrace::message` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/JournalTrace.h`

6.372 `activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller` Class Reference

Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p.1769).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/JournalTraceMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller`:

Public Member Functions

- `JournalTraceMarshaller ()`
- `virtual ~JournalTraceMarshaller ()`
- `virtual commands::DataStructure * createObject () const`

Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.372.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p.1769). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.372.2 Constructor & Destructor Documentation

6.372.2.1 `activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller::JournalTraceMarshaller () [inline]`

6.372.2.2 `virtual activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller::~~JournalTraceMarshaller () [inline, virtual]`

6.372.3 Member Function Documentation

6.372.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.372.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.372.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1342).

6.372.3.4 virtual void activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1348).

6.372.3.5 virtual int activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1354).

6.372.3.6 virtual void activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions

- IOException* if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1360).

6.372.3.7 virtual void **activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller::tightUnmarshal** (**OpenWireFormat** * *wireFormat*, **commands::DataStructure** * *dataStructure*, **decaf::io::DataInputStream** * *dataIn*, **utils::BooleanStream** * *bs*) throw (**decaf::io::IOException**) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1366).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/JournalTraceMarshaller.h`

6.373 **activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller** Class Reference

Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 1773).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/JournalTraceMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller**:

Public Member Functions

- **JournalTraceMarshaller** ()
- virtual **~JournalTraceMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.373.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p.1773). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.373.2 Constructor & Destructor Documentation

6.373.2.1 `activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller::JournalTraceMarshaller () [inline]`

6.373.2.2 `virtual activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller::~~JournalTraceMarshaller () [inline, virtual]`

6.373.3 Member Function Documentation

6.373.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.373.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.373.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1342).

6.373.3.4 virtual void activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1348).

6.373.3.5 virtual int activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1354).

6.373.3.6 virtual void activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1360).

6.373.3.7 virtual void **activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller::tightUnmarshal**
 (**OpenWireFormat** * *wireFormat*, **commands::DataStructure**
 * *dataStructure*, **decaf::io::DataInputStream** * *dataIn*,
utils::BooleanStream * *bs*) throw (**decaf::io::IOException**) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1366).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/JournalTraceMarshaller.h`

6.374 **activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller** Class Reference

Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 1777).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/JournalTraceMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller**:

Public Member Functions

- **JournalTraceMarshaller** ()
- virtual **~JournalTraceMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.374.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p.1777). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.374.2 Constructor & Destructor Documentation

6.374.2.1 `activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller::JournalTraceMarshaller () [inline]`

6.374.2.2 `virtual activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller::~~JournalTraceMarshaller () [inline, virtual]`

6.374.3 Member Function Documentation

6.374.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.374.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.374.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1342).

6.374.3.4 virtual void activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1348).

6.374.3.5 virtual int activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1354).

6.374.3.6 virtual void activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions

- IOException* if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1360).

6.374.3.7 virtual void **activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller::tightUnmarshal**
 (**OpenWireFormat** * *wireFormat*, **commands::DataStructure**
 * *dataStructure*, **decaf::io::DataInputStream** * *dataIn*,
utils::BooleanStream * *bs*) throw (**decaf::io::IOException**) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1366).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/JournalTraceMarshaller.h`

6.375 **activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller** Class Reference

Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 1781).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/JournalTraceMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller**:

Public Member Functions

- **JournalTraceMarshaller** ()
- virtual **~JournalTraceMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.375.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p.1781). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.375.2 Constructor & Destructor Documentation

6.375.2.1 `activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller::JournalTraceMarshaller () [inline]`

6.375.2.2 `virtual activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller::~~JournalTraceMarshaller () [inline, virtual]`

6.375.3 Member Function Documentation

6.375.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.375.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.375.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1342).

6.375.3.4 virtual void activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1348).

6.375.3.5 virtual int activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1354).

6.375.3.6 virtual void activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1360).

6.375.3.7 virtual void **activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller::tightUnmarshal**
 (**OpenWireFormat** * *wireFormat*, **commands::DataStructure**
 * *dataStructure*, **decaf::io::DataInputStream** * *dataIn*,
utils::BooleanStream * *bs*) throw (**decaf::io::IOException**) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1366).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/JournalTraceMarshaller.h`

6.376 **activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller** Class Reference

Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 1785).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/JournalTraceMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller**:

Public Member Functions

- **JournalTraceMarshaller** ()
- virtual **~JournalTraceMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.376.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p.1785). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.376.2 Constructor & Destructor Documentation

6.376.2.1 `activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller::JournalTraceMarshaller () [inline]`

6.376.2.2 `virtual activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller::~~JournalTraceMarshaller () [inline, virtual]`

6.376.3 Member Function Documentation

6.376.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.376.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.376.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1342).

6.376.3.4 virtual void activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1348).

6.376.3.5 virtual int activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1354).

6.376.3.6 virtual void activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1366).

```
6.376.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1366).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/JournalTraceMarshaller.h`

6.377 `activemq::commands::JournalTransaction` Class Reference

```
#include <src/main/activemq/commands/JournalTransaction.h>
```

Inheritance diagram for `activemq::commands::JournalTransaction`:

Public Member Functions

- `JournalTransaction ()`
- `virtual ~JournalTransaction ()`

- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **JournalTransaction** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1372) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **TransactionId** > & **getTransactionId** () const
- virtual **Pointer**< **TransactionId** > & **getTransactionId** ()
- virtual void **setTransactionId** (const **Pointer**< **TransactionId** > &transactionId)
- virtual unsigned char **getType** () const
- virtual void **setType** (unsigned char type)
- virtual bool **getWasPrepared** () const
- virtual void **setWasPrepared** (bool wasPrepared)

Static Public Attributes

- static const unsigned char **ID_JOURNALTRANSACTION** = 54

Protected Member Functions

- **JournalTransaction** (const **JournalTransaction** &)
- **JournalTransaction** & **operator=** (const **JournalTransaction** &)

Protected Attributes

- **Pointer**< **TransactionId** > **transactionId**
- unsigned char **type**
- bool **wasPrepared**

6.377.1 Constructor & Destructor Documentation

6.377.1.1 `activemq::commands::JournalTransaction::JournalTransaction (const JournalTransaction &) [inline, protected]`

6.377.1.2 `activemq::commands::JournalTransaction::JournalTransaction ()`

6.377.1.3 `virtual activemq::commands::JournalTransaction::~~JournalTransaction () [virtual]`

6.377.2 Member Function Documentation

6.377.2.1 `virtual JournalTransaction* activemq::commands::JournalTransaction::cloneDataStructure () const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1373).

6.377.2.2 `virtual void activemq::commands::JournalTransaction::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

Implements `activemq::commands::DataStructure` (p. 1374).

6.377.2.3 `virtual bool activemq::commands::JournalTransaction::equals (const DataStructure * value) const [virtual]`

Compares the `DataStructure` (p. 1372) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Implements `activemq::commands::DataStructure` (p. 1374).

6.377.2.4 virtual unsigned char activemq::commands::JournalTransaction::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataSet** (p. 1372) type copy.

Implements **activemq::commands::DataSet** (p. 1375).

6.377.2.5 virtual Pointer<TransactionId>& activemq::commands::JournalTransaction::getTransactionId () [virtual]

6.377.2.6 virtual const Pointer<TransactionId>& activemq::commands::JournalTransaction::getTransactionId () const [virtual]

6.377.2.7 virtual unsigned char activemq::commands::JournalTransaction::getType () const [virtual]

6.377.2.8 virtual bool activemq::commands::JournalTransaction::getWasPrepared () const [virtual]

6.377.2.9 JournalTransaction& activemq::commands::JournalTransaction::operator= (const JournalTransaction &) [inline, protected]

6.377.2.10 virtual void activemq::commands::JournalTransaction::setTransactionId (const Pointer< TransactionId > & *transactionId*) [virtual]

6.377.2.11 virtual void activemq::commands::JournalTransaction::setType (unsigned char *type*) [virtual]

6.377.2.12 virtual void activemq::commands::JournalTransaction::setWasPrepared (bool *wasPrepared*) [virtual]

6.377.2.13 virtual std::string activemq::commands::JournalTransaction::toString () const [virtual]

Returns a string containing the information for this **DataSet** (p. 1372) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataSet** (p. 662).

6.377.3 Field Documentation

- 6.377.3.1 `const unsigned char activemq::commands::JournalTransaction::ID _JOURNALTRANSACTION = 54` [static]
- 6.377.3.2 `Pointer<TransactionId> activemq::commands::JournalTransaction::transactionId` [protected]
- 6.377.3.3 `unsigned char activemq::commands::JournalTransaction::type` [protected]
- 6.377.3.4 `bool activemq::commands::JournalTransaction::wasPrepared` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/JournalTransaction.h`

6.378 `activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller` Class Reference

Marshaling code for Open Wire Format for `JournalTransactionMarshaller` (p. 1793).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/JournalTransactionMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller`:

Public Member Functions

- `JournalTransactionMarshaller ()`
- `virtual ~JournalTransactionMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaller.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.378.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p.1793). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.378.2 Constructor & Destructor Documentation

6.378.2.1 **activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller::JournalTrans**
 () [inline]

6.378.2.2 **virtual**
activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller::~~JournalTrans
 () [inline, virtual]

6.378.3 Member Function Documentation

6.378.3.1 **virtual commands::DataStructure* ac-**
tivemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller::createObject
 () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1331).

6.378.3.2 **virtual unsigned char ac-**
tivemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller::getDataStructu
 () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.378.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1342).

6.378.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1348).

6.378.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1354).

6.378.3.6 `virtual void activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1360).

6.378.3.7 `virtual void activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1366).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/JournalTransactionMarshaller.h`

6.379 activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller Class Reference

Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 1797).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/JournalTransactionMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller`:

Public Member Functions

- **JournalTransactionMarshaller** ()
- virtual **~JournalTransactionMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.379.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p.1797). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.379.2 Constructor & Destructor Documentation

6.379.2.1 activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller::JournalTransactionMarshaller () [inline]

6.379.2.2 virtual
activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller::~~JournalTransactionMarshaller () [inline, virtual]

6.379.3 Member Function Documentation

6.379.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1331).

6.379.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1336).

6.379.3.3 virtual void activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1342).

6.379.3.4 virtual void activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1348).

6.379.3.5 virtual int activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1354).

6.379.3.6 virtual void **activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller::tightMarshal2**
(**OpenWireFormat** * *wireFormat*, **commands::DataStructure**
* *dataStructure*, **decaf::io::DataOutputStream** * *dataOut*,
utils::BooleanStream * *bs*) throw (**decaf::io::IOException**) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1360).

6.379.3.7 virtual void **activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller::tightUnmarshal**
(**OpenWireFormat** * *wireFormat*, **commands::DataStructure**
* *dataStructure*, **decaf::io::DataInputStream** * *dataIn*,
utils::BooleanStream * *bs*) throw (**decaf::io::IOException**) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1366).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/JournalTransactionMarshaller.h`

6.380 **activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller** Class Reference

Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 1801).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/JournalTransactionMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller**:

Public Member Functions

- **JournalTransactionMarshaller** ()
- virtual **~JournalTransactionMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

6.380.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p.1801). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.380.2 Constructor & Destructor Documentation

6.380.2.1 `activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller::JournalTrans`
`()` [inline]

6.380.2.2 `virtual`
`activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller::~~JournalTran`
`()` [inline, virtual]

6.380.3 Member Function Documentation

6.380.3.1 `virtual commands::DataStructure* ac-`
`tivemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller::createObject`
`() const` [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p.1331).

6.380.3.2 `virtual unsigned char ac-`
`tivemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller::getDataStructur`
`() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p.1336).

6.380.3.3 `virtual void ac-`
`tivemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller::looseMarshal`
`(OpenWireFormat * wireFormat, commands::DataStructure *`
`dataStructure, decaf::io::DataOutputStream * dataOut) throw (`
`decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1342).

```
6.380.3.4 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller::looseUnmarsha
l ( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1348).

```
6.380.3.5 virtual int ac-
tivemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller::tightMarshall
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, utils::BooleanStream * bs ) throw (
decaf::io::IOException ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1354).

6.380.3.6 virtual void activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1360).

6.380.3.7 virtual void activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1366).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**JournalTransactionMarshaller.h**

6.381 activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller Class Reference

Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 1804).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/JournalTransactionMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller`:

Public Member Functions

- **JournalTransactionMarshaller** ()
- virtual **~JournalTransactionMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.381.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p.1804). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module

6.381.2 Constructor & Destructor Documentation

6.381.2.1 `activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller::JournalTrans`
() [inline]

6.381.2.2 `virtual`
`activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller::~~JournalTran`
() [inline, virtual]

6.381.3 Member Function Documentation

6.381.3.1 `virtual commands::DataStructure* ac-`
`tivemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller::createObject`
() const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.381.3.2 `virtual unsigned char ac-`
`tivemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller::getDataStructu`
() const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.381.3.3 `virtual void ac-`
`tivemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller::looseMarshal`
(`OpenWireFormat * wireFormat`, `commands::DataStructure * dataStructure`, `decaf::io::DataOutputStream * dataOut`) throw (`decaf::io::IOException`) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1342).

6.381.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1348).

6.381.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1354).

6.381.3.6 `virtual void activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1360).

6.381.3.7 `virtual void activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller::tightUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1366).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/JournalTransactionMarshaller.h`

6.382 activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller Class Reference

Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 1808).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/JournalTransactionMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller**:

Public Member Functions

- **JournalTransactionMarshaller** ()
- virtual **~JournalTransactionMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.382.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p.1808). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.382.2 Constructor & Destructor Documentation

6.382.2.1 `activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller::JournalTrans`
`() [inline]`

6.382.2.2 `virtual`
`activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller::~~JournalTra`
`() [inline, virtual]`

6.382.3 Member Function Documentation

6.382.3.1 `virtual commands::DataStructure* ac-`
`tivemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller::createObject`
`() const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.382.3.2 `virtual unsigned char ac-`
`tivemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller::getDataStructu`
`() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.382.3.3 `virtual void ac-`
`tivemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller::looseMarshal`
`(OpenWireFormat * wireFormat, commands::DataStructure *`
`dataStructure, decaf::io::DataOutputStream * dataOut) throw (`
`decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1342).

6.382.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1348).

6.382.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1354).

6.382.3.6 `virtual void activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions

- IOException* if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1360).

6.382.3.7 virtual void **activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller::tightUnmarshal**
 (**OpenWireFormat** * *wireFormat*, **commands::DataStructure**
 * *dataStructure*, **decaf::io::DataInputStream** * *dataIn*,
utils::BooleanStream * *bs*) throw (**decaf::io::IOException**) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1366).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/JournalTransactionMarshaller.h`

6.383 activemq::commands::KeepAliveInfo Class Reference

```
#include <src/main/activemq/commands/KeepAliveInfo.h>
```

Inheritance diagram for **activemq::commands::KeepAliveInfo**:

Public Member Functions

- **KeepAliveInfo** ()
- virtual **~KeepAliveInfo** ()
- virtual unsigned char **getDataStructureType** () const

Get the unique identifier that this object and its own Marshaler share.

- virtual **KeepAliveInfo** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1372) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent.*
- virtual bool **isKeepAliveInfo** () const
- virtual **Pointer**< **Command** > **visit** (**activemq::state::CommandVisitor** *visitor) throw (exceptions::ActiveMQException)
Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_KEEPAVAILABLEINFO** = 10

Protected Member Functions

- **KeepAliveInfo** (const **KeepAliveInfo** &)
- **KeepAliveInfo** & **operator=** (const **KeepAliveInfo** &)

6.383.1 Constructor & Destructor Documentation

- 6.383.1.1 **activemq::commands::KeepAliveInfo::KeepAliveInfo** (const **KeepAliveInfo** &) [inline, protected]
- 6.383.1.2 **activemq::commands::KeepAliveInfo::KeepAliveInfo** ()
- 6.383.1.3 **virtual activemq::commands::KeepAliveInfo::~~KeepAliveInfo** () [virtual]

6.383.2 Member Function Documentation

- 6.383.2.1 **virtual KeepAliveInfo*** **activemq::commands::KeepAliveInfo::cloneDataStructure** () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1373).

6.383.2.2 virtual void activemq::commands::KeepAliveInfo::copyDataStructure (const DataStructure * *src*) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 598).

6.383.2.3 virtual bool activemq::commands::KeepAliveInfo::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 598).

6.383.2.4 virtual unsigned char activemq::commands::KeepAliveInfo::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataStructure** (p. 1372) type copy.

Implements **activemq::commands::DataStructure** (p. 1375).

6.383.2.5 virtual bool activemq::commands::KeepAliveInfo::isKeepAliveInfo () const [inline, virtual]**Returns**

an answer of true to the **isKeepAliveInfo()** (p. 1814) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 600).

6.383.2.6 `KeepAliveInfo& activemq::commands::KeepAliveInfo::operator= (const KeepAliveInfo &) [inline, protected]`

6.383.2.7 `virtual std::string activemq::commands::KeepAliveInfo::toString () const [virtual]`

Returns a string containing the information for this **DataStructure** (p.1372) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from `activemq::commands::BaseCommand` (p. 602).

6.383.2.8 `virtual Pointer<Command> activemq::commands::KeepAliveInfo::visit (activemq::state::CommandVisitor * visitor) throw (exceptions::ActiveMQException) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

Returns

a **Response** (p. 2611) to the visitor being called or NULL if no response.

Implements `activemq::commands::Command` (p.995).

6.383.3 Field Documentation

6.383.3.1 `const unsigned char activemq::commands::KeepAliveInfo::ID _ - KEEPALIVEINFO = 10 [static]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/KeepAliveInfo.h`

6.384 activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p.1815).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/KeepAliveInfoMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller`:

Public Member Functions

- **KeepAliveInfoMarshaller** ()
- virtual **~KeepAliveInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.384.1 Detailed Description

Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p.1815). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.384.2 Constructor & Destructor Documentation

6.384.2.1 `activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller::KeepAliveInfoMar`
`() [inline]`

6.384.2.2 `virtual`
`activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller::~~KeepAliveInfoM`
`() [inline, virtual]`

6.384.3 Member Function Documentation

6.384.3.1 `virtual commands::DataStructure* ac-`
`tivemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller::createObject`
`() const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.384.3.2 `virtual unsigned char ac-`
`tivemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller::getDataStructureTy`
`() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.384.3.3 `virtual void ac-`
`tivemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller::looseMarshal`
`(OpenWireFormat * wireFormat, commands::DataStructure *`
`dataStructure, decaf::io::DataOutputStream * dataOut) throw (`
`decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 631).

6.384.3.4 virtual void activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 632).

6.384.3.5 virtual int activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 633).

6.384.3.6 virtual void activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 634).

```
6.384.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 635).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/KeepAliveInfoMarshaller.h`

6.385 `activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller` Class Reference

Marshaling code for Open Wire Format for `KeepAliveInfoMarshaller` (p. 1819).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/KeepAliveInfoMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller`:

Public Member Functions

- **KeepAliveInfoMarshaller** ()
- virtual **~KeepAliveInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.385.1 Detailed Description

Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p.1819). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.385.2 Constructor & Destructor Documentation

6.385.2.1 `activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller::KeepAliveInfoMar
() [inline]`

6.385.2.2 `virtual
activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller::~~KeepAliveInfoM
() [inline, virtual]`

6.385.3 Member Function Documentation

6.385.3.1 `virtual commands::DataStructure* ac-
tivemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller::createObject
() const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.385.3.2 `virtual unsigned char ac-
tivemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller::getDataStructureTy
() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.385.3.3 `virtual void ac-
tivemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller::looseMarshal
(OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut) throw (
decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 624).

6.385.3.4 virtual void activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 625).

6.385.3.5 virtual int activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 626).

6.385.3.6 virtual void activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 628).

```
6.385.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 629).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/KeepAliveInfoMarshaller.h`

6.386 `activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller` Class Reference

Marshaling code for Open Wire Format for `KeepAliveInfoMarshaller` (p. 1823).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/KeepAliveInfoMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller`:

Public Member Functions

- **KeepAliveInfoMarshaller** ()
- virtual **~KeepAliveInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.386.1 Detailed Description

Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p.1823). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.386.2 Constructor & Destructor Documentation

6.386.2.1 `activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller::KeepAliveInfoMar
() [inline]`

6.386.2.2 `virtual
activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller::~~KeepAliveInfoM
() [inline, virtual]`

6.386.3 Member Function Documentation

6.386.3.1 `virtual commands::DataStructure* ac-
tivemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller::createObject
() const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.386.3.2 `virtual unsigned char ac-
tivemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller::getDataStructureTy
() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.386.3.3 `virtual void ac-
tivemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller::looseMarshal
(OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut) throw (
decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 604).

6.386.3.4 virtual void activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 605).

6.386.3.5 virtual int activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 606).

6.386.3.6 virtual void activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 608).

```
6.386.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 609).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/KeepAliveInfoMarshaller.h`

6.387 `activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller` Class Reference

Marshaling code for Open Wire Format for `KeepAliveInfoMarshaller` (p. 1827).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/KeepAliveInfoMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller`:

Public Member Functions

- **KeepAliveInfoMarshaller** ()
- virtual **~KeepAliveInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.387.1 Detailed Description

Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p.1827). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.387.2 Constructor & Destructor Documentation

6.387.2.1 `activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller::KeepAliveInfoMar
() [inline]`

6.387.2.2 `virtual
activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller::~~KeepAliveInfoM
() [inline, virtual]`

6.387.3 Member Function Documentation

6.387.3.1 `virtual commands::DataStructure* ac-
tivemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller::createObject
() const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.387.3.2 `virtual unsigned char ac-
tivemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller::getDataStructureTy
() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.387.3.3 `virtual void ac-
tivemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller::looseMarshal
(OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut) throw (
decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 618).

6.387.3.4 virtual void activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 619).

6.387.3.5 virtual int activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 620).

6.387.3.6 virtual void activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 621).

```
6.387.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 622).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/KeepAliveInfoMarshaller.h`

6.388 `activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller` Class Reference

Marshaling code for Open Wire Format for `KeepAliveInfoMarshaller` (p. 1831).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/KeepAliveInfoMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller`:

Public Member Functions

- **KeepAliveInfoMarshaller** ()
- virtual **~KeepAliveInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.388.1 Detailed Description

Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p.1831). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.388.2 Constructor & Destructor Documentation

6.388.2.1 `activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller::KeepAliveInfoMar`
`() [inline]`

6.388.2.2 `virtual`
`activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller::~~KeepAliveInfoM`
`() [inline, virtual]`

6.388.3 Member Function Documentation

6.388.3.1 `virtual commands::DataStructure* ac-`
`tivemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller::createObject`
`() const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.388.3.2 `virtual unsigned char ac-`
`tivemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller::getDataStructureTy`
`() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.388.3.3 `virtual void ac-`
`tivemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller::looseMarshal`
`(OpenWireFormat * wireFormat, commands::DataStructure *`
`dataStructure, decaf::io::DataOutputStream * dataOut) throw (`
`decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 611).

6.388.3.4 virtual void activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 612).

6.388.3.5 virtual int activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 613).

6.388.3.6 virtual void activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 614).

```
6.388.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 615).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/KeepAliveInfoMarshaller.h`

6.389 decaf::security::Key Class Reference

The **Key** (p. 1835) interface is the top-level interface for all keys.

```
#include <src/main/decaf/security/Key.h>
```

Inheritance diagram for `decaf::security::Key`:

Public Member Functions

- virtual `~Key ()`
- virtual `std::string getAlgorithm () const =0`

Returns the standard algorithm name for this key.

- virtual void `getEncoded (std::vector< unsigned char > &output) const =0`

Provides the key in its primary encoding format, or nothing if this key does not support encoding.

- virtual `std::string getFormat () const =0`

Returns the name of the primary encoding format of this key, or an empty string if this key does not support encoding.

6.389.1 Detailed Description

The **Key** (p.1835) interface is the top-level interface for all keys. It defines the functionality shared by all key objects. All keys have three characteristics:

An Algorithm

This is the key algorithm for that key. The key algorithm is usually an encryption or asymmetric operation algorithm (such as DSA or RSA), which will work with those algorithms and with related algorithms (such as MD5 with RSA, SHA-1 with RSA, Raw DSA, etc.) The name of the algorithm of a key is obtained using the `getAlgorithm` method.

An Encoded Form

This is an external encoded form for the key used when a standard representation of the key is needed outside the application, as when transmitting the key to some other party. The key is encoded according to a standard format (such as X.509 `SubjectPublicKeyInfo` or PKCS#8), and is returned using the `getEncoded` method. Note: The syntax of the ASN.1 type `SubjectPublicKeyInfo` is defined as follows:

```
SubjectPublicKeyInfo ::= SEQUENCE {
    algorithm AlgorithmIdentifier,
    subjectPublicKey BIT STRING }
AlgorithmIdentifier ::= SEQUENCE {
    algorithm OBJECT IDENTIFIER,
    parameters ANY DEFINED BY algorithm OPTIONAL }
```

For more information, see RFC 2459: Internet X.509 Public **Key** (p.1835) Infrastructure Certificate and CRL Profile.

A Format

This is the name of the format of the encoded key. It is returned by the `getFormat` method.

6.389.2 Constructor & Destructor Documentation

6.389.2.1 `virtual decaf::security::Key::~~Key () [inline, virtual]`

6.389.3 Member Function Documentation

6.389.3.1 `virtual std::string decaf::security::Key::getAlgorithm () const [pure virtual]`

Returns the standard algorithm name for this key.

For example, "DSA" would indicate that this key is a DSA key.

Returns

the name of the algorithm associated with this key.

6.389.3.2 `virtual void decaf::security::Key::getEncoded (std::vector< unsigned char > & output) const [pure virtual]`

Provides the key in its primary encoding format, or nothing if this key does not support encoding.

Parameters

output Receives the encoded key, or nothing if the key does not support encoding.

6.389.3.3 `virtual std::string decaf::security::Key::getFormat () const [pure virtual]`

Returns the name of the primary encoding format of this key, or an empty string if this key does not support encoding.

The primary encoding format is named in terms of the appropriate ASN.1 data format, if an ASN.1 specification for this key exists. For example, the name of the ASN.1 data format for public keys is SubjectPublicKeyInfo, as defined by the X.509 standard; in this case, the returned format is "X.509". Similarly, the name of the ASN.1 data format for private keys is PrivateKeyInfo, as defined by the PKCS #8 standard; in this case, the returned format is "PKCS#8".

Returns

the primary encoding format of the key.

The documentation for this class was generated from the following file:

- `src/main/decaf/security/Key.h`

6.390 decaf::security::KeyException Class Reference

```
#include <src/main/decaf/security/KeyException.h>
```

Inheritance diagram for decaf::security::KeyException:

Public Member Functions

- **KeyException** () throw ()
Default Constructor.
- **KeyException** (const Exception &ex) throw ()
Conversion Constructor from some other Exception.
- **KeyException** (const **KeyException** &ex) throw ()
Copy Constructor.
- **KeyException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **KeyException** (const std::exception *cause) throw ()
Constructor.
- **KeyException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **KeyException** * **clone** () const
Clones this exception.
- virtual ~**KeyException** () throw ()

6.390.1 Constructor & Destructor Documentation

6.390.1.1 decaf::security::KeyException::KeyException () throw () [inline]

Default Constructor.

6.390.1.2 decaf::security::KeyException::KeyException (const Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

Parameters

ex An exception that should become this type of Exception

6.390.1.3 decaf::security::KeyException::KeyException (const KeyException & ex) throw () [inline]

Copy Constructor.

Parameters

ex An exception that should become this type of Exception

6.390.1.4 `decaf::security::KeyException::KeyException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.390.1.5 `decaf::security::KeyException::KeyException (const std::exception * cause) throw () [inline]`

Constructor.

Parameters

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.390.1.6 `decaf::security::KeyException::KeyException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file name where exception occurs

lineNumber line number where the exception occurred.

msg message to report

... list of primitives that are formatted into the message

6.390.1.7 `virtual decaf::security::KeyException::~~KeyException () throw () [inline, virtual]`

6.390.2 Member Function Documentation

6.390.2.1 `virtual KeyException* decaf::security::KeyException::clone () const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

A deep copy of this exception.

Reimplemented from **decaf::security::GeneralSecurityException** (p. 1604).

Reimplemented in **decaf::security::InvalidKeyException** (p. 1700).

The documentation for this class was generated from the following file:

- `src/main/decaf/security/KeyException.h`

6.391 activemq::commands::LastPartialCommand Class Reference

```
#include <src/main/activemq/commands/LastPartialCommand.h>
```

Inheritance diagram for `activemq::commands::LastPartialCommand`:

Public Member Functions

- **LastPartialCommand** ()
- virtual **~LastPartialCommand** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **LastPartialCommand * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1372) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent.*

Static Public Attributes

- static const unsigned char **ID_LASTPARTIALCOMMAND** = 61

Protected Member Functions

- **LastPartialCommand** (const **LastPartialCommand** &)
- **LastPartialCommand** & **operator=** (const **LastPartialCommand** &)

6.391.1 Constructor & Destructor Documentation

- 6.391.1.1** `activemq::commands::LastPartialCommand::LastPartialCommand (const LastPartialCommand &) [inline, protected]`
- 6.391.1.2** `activemq::commands::LastPartialCommand::LastPartialCommand ()`
- 6.391.1.3** `virtual
activemq::commands::LastPartialCommand::~~LastPartialCommand ()
[virtual]`

6.391.2 Member Function Documentation

- 6.391.2.1** `virtual LastPartialCommand* ac-
tivemq::commands::LastPartialCommand::cloneDataStructure () const
[virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Reimplemented from `activemq::commands::PartialCommand` (p. 2320).

- 6.391.2.2** `virtual void ac-
tivemq::commands::LastPartialCommand::copyDataStructure (const
DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

Reimplemented from `activemq::commands::PartialCommand` (p. 2320).

- 6.391.2.3** `virtual bool activemq::commands::LastPartialCommand::equals (const
DataStructure * value) const [virtual]`

Compares the `DataStructure` (p. 1372) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::PartialCommand` (p. 2320).

6.391.2.4 `virtual unsigned char activemq::commands::LastPartialCommand::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaller share.

Returns

new **DataSet** (p. 1372) type copy.

Reimplemented from **activemq::commands::PartialCommand** (p. 2321).

6.391.2.5 `LastPartialCommand& activemq::commands::LastPartialCommand::operator= (const LastPartialCommand &) [inline, protected]`

6.391.2.6 `virtual std::string activemq::commands::LastPartialCommand::toString () const [virtual]`

Returns a string containing the information for this **DataSet** (p. 1372) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::PartialCommand** (p. 2321).

6.391.3 Field Documentation

6.391.3.1 `const unsigned char activemq::commands::LastPartialCommand::ID__LASTPARTIALCOMMAND = 61 [static]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/LastPartialCommand.h`

6.392 activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 1842).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/LastPartialCommandMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller**:

Public Member Functions

- **LastPartialCommandMarshaller** ()
- virtual **~LastPartialCommandMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.392.1 Detailed Description

Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p.1842).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.392.2 Constructor & Destructor Documentation

6.392.2.1 `activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller::LastPartialCommandMarshaller()` `[inline]`

6.392.2.2 `virtual`
`activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller::~LastPartialCommandMarshaller()` `[inline, virtual]`

6.392.3 Member Function Documentation

6.392.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller::createObject()` `const` `[virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller` (p. 2323).

6.392.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller::getDataStructureType()` `const` `[virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller` (p. 2323).

6.392.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` `[virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller` (p. 2324).

6.392.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller` (p. 2324).

6.392.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller` (p. 2325).

6.392.3.6 virtual void activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller::tightMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller** (p. 2325).

6.392.3.7 virtual void activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller** (p. 2326).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**LastPartialCommandMarshaller.h**

6.393 activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 1846).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/LastPartialCommandMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller`:

Public Member Functions

- **LastPartialCommandMarshaller** ()
- virtual **~LastPartialCommandMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.393.1 Detailed Description

Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p.1846).
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module

6.393.2 Constructor & Destructor Documentation

6.393.2.1 `activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller::LastPartialCommandMarshaller()` [inline]

6.393.2.2 `virtual activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller::~~LastPartialCommandMarshaller()` [inline, virtual]

6.393.3 Member Function Documentation

6.393.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller` (p. 2336).

6.393.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller` (p. 2336).

6.393.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller` (p. 2336).

6.393.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller` (p. 2337).

6.393.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller` (p. 2337).

6.393.3.6 virtual void activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller::tightMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller** (p. 2338).

6.393.3.7 virtual void activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller** (p. 2338).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**LastPartialCommandMarshaller.h**

6.394 activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 1850).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/LastPartialCommandMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller`:

Public Member Functions

- **LastPartialCommandMarshaller** ()
- virtual **~LastPartialCommandMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.394.1 Detailed Description

Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p.1850).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module

6.394.2 Constructor & Destructor Documentation

6.394.2.1 `activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller::LastPartialCommandMarshaller()` [inline]

6.394.2.2 `virtual activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller::~~LastPartialCommandMarshaller()` [inline, virtual]

6.394.3 Member Function Documentation

6.394.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller` (p. 2327).

6.394.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller` (p. 2328).

6.394.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller` (p. 2328).

6.394.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller` (p. 2328).

6.394.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller` (p. 2329).

6.394.3.6 virtual void activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller::tightMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller** (p. 2329).

6.394.3.7 virtual void activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller** (p. 2330).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/LastPartialCommandMarshaller.h

6.395 activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 1854).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/LastPartialCommandMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller`:

Public Member Functions

- **LastPartialCommandMarshaller** ()
- virtual **~LastPartialCommandMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.395.1 Detailed Description

Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p.1854).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module

6.395.2 Constructor & Destructor Documentation

6.395.2.1 `activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller::LastPartialCommandMarshaller()` [inline]

6.395.2.2 `virtual activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller::~~LastPartialCommandMarshaller()` [inline, virtual]

6.395.3 Member Function Documentation

6.395.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller::createObject(const)` [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller` (p. 2332).

6.395.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller` (p. 2332).

6.395.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut)` throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller` (p. 2332).

6.395.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller` (p. 2333).

6.395.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller` (p. 2333).

6.395.3.6 virtual void activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller::tightMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller** (p. 2334).

6.395.3.7 virtual void activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller** (p. 2334).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**LastPartialCommandMarshaller.h**

6.396 activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 1858).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/LastPartialCommandMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller`:

Public Member Functions

- **LastPartialCommandMarshaller** ()
- virtual **~LastPartialCommandMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.396.1 Detailed Description

Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p.1858).
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the `activemq-openwire-generator` module

6.396.2 Constructor & Destructor Documentation

6.396.2.1 `activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller::LastPartialCommandMarshaller()` [inline]

6.396.2.2 `virtual activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller::~~LastPartialCommandMarshaller()` [inline, virtual]

6.396.3 Member Function Documentation

6.396.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller` (p. 2340).

6.396.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Reimplemented from `activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller` (p. 2340).

6.396.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller` (p. 2341).

6.396.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller` (p. 2341).

6.396.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller::tightMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller` (p. 2341).

6.396.3.6 virtual void activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller::tightMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller** (p. 2342).

6.396.3.7 virtual void activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller** (p. 2342).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/LastPartialCommandMarshaller.h

6.397 decaf::util::comparators::Less< E > Class Template Reference

Simple **Less** (p. 1862) **Comparator** (p. 1011) that compares to elements to determine if the first is less than the second.

```
#include <src/main/decaf/util/comparators/Less.h>
```

Inheritance diagram for decaf::util::comparators::Less< E >:

Public Member Functions

- **Less** ()
- virtual **~Less** ()
- virtual bool **operator**() (const E &left, const E &right) const
*Implementation of the Binary function interface as a means of allowing a **Comparator** (p. 1011) to be passed to an STL **Map** (p. 1970) for use as the sorting criteria.*
- virtual int **compare** (const E &o1, const E &o2) const
Compares its two arguments for order.

6.397.1 Detailed Description

```
template<typename E> class decaf::util::comparators::Less< E >
```

Simple **Less** (p. 1862) **Comparator** (p. 1011) that compares to elements to determine if the first is less than the second. This can be used in **Collection** (p. 982) classes to sort elements according to their natural ordering. By design the Comparator's compare function return more information about comparison than the STL binary function's boolean compare operator. In this case the compare method will return

Since

1.0

6.397.2 Constructor & Destructor Documentation

6.397.2.1 template<typename E > decaf::util::comparators::Less< E >::Less ()
 [inline]

6.397.2.2 template<typename E > virtual decaf::util::comparators::Less< E
 >::~~Less () [inline, virtual]

6.397.3 Member Function Documentation

6.397.3.1 template<typename E > virtual int decaf::util::comparators::Less< E
 >::compare (const E & o1, const E & o2) const [inline, virtual]

Compares its two arguments for order.

Returns a negative integer, zero, or a positive integer as the first argument is less than, equal to, or greater than the second.

The implementor must ensure that `sgn(compare(x, y)) == -sgn(compare(y, x))` for all x and y. (This implies that `compare(x, y)` must throw an exception if and only if `compare(y, x)` throws an exception.)

The implementor must also ensure that the relation is transitive: `((compare(x, y)>0) && (compare(y, z)>0))` implies `compare(x, z)>0`.

Finally, the implementor must ensure that `compare(x, y)==0` implies that `sgn(compare(x, z))==sgn(compare(y, z))` for all `z`.

It is generally the case, but not strictly required that `(compare(x, y)==0) == (x == y)`. Generally speaking, any comparator that violates this condition should clearly indicate this fact. The recommended language is "Note: this comparator imposes orderings that are inconsistent with equals."

Parameters

- o1* - the first object to be compared
- o2* - the second object to be compared

Returns

a negative integer, zero, or a positive integer as the first argument is less than, equal to, or greater than the second.

Implements `decaf::util::Comparator< E >` (p. 1012).

6.397.3.2 `template<typename E> virtual bool decaf::util::comparators::Less< E >::operator() (const E & left, const E & right) const` [`inline`, `virtual`]

Implementation of the Binary function interface as a means of allowing a **Comparator** (p. 1011) to be passed to an STL **Map** (p. 1970) for use as the sorting criteria.

Parameters

- left* - the Left hand side operand.
- right* - the Right hand side operand.

Returns

true if the vale of left is less than the value of right.

Implements `decaf::util::Comparator< E >` (p. 1013).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/comparators/Less.h`

6.398 `std::less< decaf::lang::Pointer< T > >` Struct Template Reference

An override of the less function object so that the Pointer objects can be stored in STL Maps, etc.

`#include <src/main/decaf/lang/Pointer.h>`

Inheritance diagram for `std::less< decaf::lang::Pointer< T > >`:

Public Member Functions

- bool **operator()** (const **decaf::lang::Pointer**< T > &left, const **decaf::lang::Pointer**< T > &right) const

6.398.1 Detailed Description

template<typename T> struct std::less< decaf::lang::Pointer< T > >

An override of the less function object so that the Pointer objects can be stored in STL Maps, etc.

6.398.2 Member Function Documentation

6.398.2.1 **template<typename T > bool std::less< decaf::lang::Pointer< T > >::operator()** (const **decaf::lang::Pointer**< T > & *left*, const **decaf::lang::Pointer**< T > & *right*) const [inline]

References **decaf::lang::Pointer**< T, **REFCOUNTER** >::get().

The documentation for this struct was generated from the following file:

- **src/main/decaf/lang/Pointer.h**

6.399 decaf::util::List< E > Class Template Reference

An ordered collection (also known as a sequence).

#include <src/main/decaf/util/List.h>

Inheritance diagram for **decaf::util::List**< E >:

Public Member Functions

- virtual **~List** ()
- virtual **ListIterator**< E > * **listIterator** ()=0
- virtual **ListIterator**< E > * **listIterator** () const =0
- virtual **ListIterator**< E > * **listIterator** (std::size_t index)=0 throw (**decaf::lang::exceptions::IndexOutOfBoundsException**)
- virtual **ListIterator**< E > * **listIterator** (std::size_t index) const =0 throw (**decaf::lang::exceptions::IndexOutOfBoundsException**)
- virtual std::size_t **indexOf** (const E &value)=0 throw (**decaf::lang::exceptions::NoSuchElementException**)

Returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element.

- virtual std::size_t **lastIndexOf** (const E &value)=0 throw (**decaf::lang::exceptions::NoSuchElementException**)

Returns the index of the last occurrence of the specified element in this list, or -1 if this list does not contain the element.

- virtual E **get** (std::size_t index) const =0 throw (decaf::lang::exceptions::IndexOutOfBoundsException)

Gets the element contained at position passed.

- virtual E **set** (std::size_t index, const E &element)=0 throw (decaf::lang::exceptions::IndexOutOfBoundsException)

Replaces the element at the specified position in this list with the specified element.

- virtual void **add** (std::size_t index, const E &element)=0 throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IndexOutOfBoundsException)

Inserts the specified element at the specified position in this list.

- virtual bool **addAll** (std::size_t index, const **Collection**< E > &source)=0 throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IndexOutOfBoundsException)

Inserts all of the elements in the specified collection into this list at the specified position (optional operation).

- virtual E **remove** (std::size_t index)=0 throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IndexOutOfBoundsException)

Removes the element at the specified position in this list.

6.399.1 Detailed Description

```
template<typename E> class decaf::util::List< E >
```

An ordered collection (also known as a sequence). The user of this interface has precise control over where in the list each element is inserted. The user can access elements by their integer index (position in the list), and search for elements in the list.

Unlike sets, lists typically allow duplicate elements. More formally, lists typically allow pairs of elements *e1* and *e2* such that *e1.equals(e2)*, and they typically allow multiple null elements if they allow null elements at all. It is not inconceivable that someone might wish to implement a list that prohibits duplicates, by throwing runtime exceptions when the user attempts to insert them, but we expect this usage to be rare.

6.399.2 Constructor & Destructor Documentation

6.399.2.1 `template<typename E> virtual decaf::util::List< E >::~~List ()`
`[inline, virtual]`

6.399.3 Member Function Documentation

6.399.3.1 `template<typename E> virtual void decaf::util::List< E >::add (std::size_t index, const E & element) throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IndexOutOfBoundsException) [pure virtual]`

Inserts the specified element at the specified position in this list.

Shifts the element currently at that position (if any) and any subsequent elements to the right (adds one to their indices).

Parameters

index - index at which the specified element is to be inserted

element - element to be inserted

Exceptions

IndexOutOfBoundsException - if the index is greater than size

UnsupportedOperationException - If the collection is non-modifiable.

Implemented in `decaf::util::StlList< E >` (p.2839), `decaf::util::StlList< CompositeTask * >` (p.2839), `decaf::util::StlList< URI >` (p.2839), `decaf::util::StlList< Pointer< DestinationInfo > >` (p.2839), `decaf::util::StlList< PrimitiveValueNode >` (p.2839), `decaf::util::StlList< Pointer< Command > >` (p.2839), and `decaf::util::StlList< Pointer< BackupTransport > >` (p.2839).

6.399.3.2 `template<typename E> virtual bool decaf::util::List< E >::addAll (std::size_t index, const Collection< E > & source) throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IndexOutOfBoundsException) [pure virtual]`

Inserts all of the elements in the specified collection into this list at the specified position (optional operation).

Shifts the element currently at that position (if any) and any subsequent elements to the right (increases their indices). The new elements will appear in this list in the order that they are returned by the specified collection's iterator. The behavior of this operation is undefined if the specified collection is modified while the operation is in progress. (Note that this will occur if the specified collection is this list, and it's nonempty.)

Parameters

index The index at which to insert the first element from the specified collection

source The **Collection** (p.982) containing elements to be added to this list

Returns

true if this list changed as a result of the call

Exceptions

IndexOutOfBoundsException - if the index is greater than size

UnsupportedOperationException - If the collection is non-modifiable.

Implemented in `decaf::util::StlList< E >` (p. 2839), `decaf::util::StlList< CompositeTask * >` (p. 2839), `decaf::util::StlList< URI >` (p. 2839), `decaf::util::StlList< Pointer< DestinationInfo > >` (p. 2839), `decaf::util::StlList< PrimitiveValueNode >` (p. 2839), `decaf::util::StlList< Pointer< Command > >` (p. 2839), and `decaf::util::StlList< Pointer< BackupTransport > >` (p. 2839).

6.399.3.3 `template<typename E> virtual E decaf::util::List< E >::get (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException)` [pure virtual]

Gets the element contained at position passed.

Parameters

index - position to get

Returns

value at index

Implemented in `decaf::util::StlList< E >` (p. 2841), `decaf::util::StlList< CompositeTask * >` (p. 2841), `decaf::util::StlList< URI >` (p. 2841), `decaf::util::StlList< Pointer< DestinationInfo > >` (p. 2841), `decaf::util::StlList< PrimitiveValueNode >` (p. 2841), `decaf::util::StlList< Pointer< Command > >` (p. 2841), and `decaf::util::StlList< Pointer< BackupTransport > >` (p. 2841).

6.399.3.4 `template<typename E> virtual std::size_t decaf::util::List< E >::indexOf (const E & value) throw (decaf::lang::exceptions::NoSuchElementException)` [pure virtual]

Returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element.

More formally, returns the lowest index *i* such that `get(i) == value`, or -1 if there is no such index.

Parameters

value - element to search for

Returns

the index of the first occurrence of the specified element in this list,

Exceptions

NoSuchElementException if value is not in the list

Implemented in `decaf::util::StlList< E >` (p. 2841), `decaf::util::StlList< CompositeTask * >` (p. 2841), `decaf::util::StlList< URI >` (p. 2841), `decaf::util::StlList< Pointer< DestinationInfo > >` (p. 2841), `decaf::util::StlList< PrimitiveValueNode >` (p. 2841), `decaf::util::StlList< Pointer< Command > >` (p. 2841), and `decaf::util::StlList< Pointer< BackupTransport > >` (p. 2841).

6.399.3.5 `template<typename E> virtual size_t decaf::util::List< E >::lastIndexOf (const E & value) throw (decaf::lang::exceptions::NoSuchElementException) [pure virtual]`

Returns the index of the last occurrence of the specified element in this list, or -1 if this list does not contain the element.

More formally, returns the highest index *i* such that `get(i) == value` or -1 if there is no such index.

Parameters

value - element to search for

Returns

the index of the last occurrence of the specified element in this list.

Exceptions

NoSuchElementException if *value* is not in the list

Implemented in `decaf::util::StlList< E >` (p. 2842), `decaf::util::StlList< CompositeTask * >` (p. 2842), `decaf::util::StlList< URI >` (p. 2842), `decaf::util::StlList< Pointer< DestinationInfo > >` (p. 2842), `decaf::util::StlList< PrimitiveValueNode >` (p. 2842), `decaf::util::StlList< Pointer< Command > >` (p. 2842), and `decaf::util::StlList< Pointer< BackupTransport > >` (p. 2842).

6.399.3.6 `template<typename E> virtual ListIterator<E>* decaf::util::List< E >::listIterator () [pure virtual]`

Returns

a list iterator over the elements in this list (in proper sequence).

Implemented in `decaf::util::StlList< E >` (p. 2843), `decaf::util::StlList< CompositeTask * >` (p. 2843), `decaf::util::StlList< URI >` (p. 2843), `decaf::util::StlList< Pointer< DestinationInfo > >` (p. 2843), `decaf::util::StlList< PrimitiveValueNode >` (p. 2843), `decaf::util::StlList< Pointer< Command > >` (p. 2843), and `decaf::util::StlList< Pointer< BackupTransport > >` (p. 2843).

6.399.3.7 `template<typename E> virtual ListIterator<E>* decaf::util::List< E >::listIterator (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException) [pure virtual]`

Implemented in `decaf::util::StlList< E >` (p. 2843), `decaf::util::StlList< CompositeTask * >` (p. 2843), `decaf::util::StlList< URI >` (p. 2843), `decaf::util::StlList< Pointer< DestinationInfo > >` (p. 2843), `decaf::util::StlList< PrimitiveValueNode >` (p. 2843), `decaf::util::StlList< Pointer< Command > >` (p. 2843), and `decaf::util::StlList< Pointer< BackupTransport > >` (p. 2843).

6.399.3.8 `template<typename E> virtual ListIterator<E>* decaf::util::List< E >::listIterator () const [pure virtual]`

Implemented in `decaf::util::StlList< E >` (p. 2843), `decaf::util::StlList< CompositeTask * >` (p. 2843), `decaf::util::StlList< URI >` (p. 2843), `decaf::util::StlList< Pointer<`

DestinationInfo > > (p. 2843), **decaf::util::StlList< PrimitiveValueNode >** (p. 2843), **decaf::util::StlList< Pointer< Command > >** (p. 2843), and **decaf::util::StlList< Pointer< BackupTransport > >** (p. 2843).

6.399.3.9 `template<typename E> virtual ListIterator<E>* decaf::util::List< E >::listIterator (std::size_t index) throw (decaf::lang::exceptions::IndexOutOfBoundsException)` [pure virtual]

Parameters

index index of first element to be returned from the list iterator (by a call to the next method).

Returns

a list iterator of the elements in this list (in proper sequence), starting at the specified position in this list. The specified index indicates the first element that would be returned by an initial call to next. An initial call to previous would return the element with the specified index minus one.

Exceptions

IndexOutOfBoundsException if the index is out of range (`index < 0 || index > size()` (p. 990))

Implemented in **decaf::util::StlList< E >** (p. 2842), **decaf::util::StlList< CompositeTask * >** (p. 2842), **decaf::util::StlList< URI >** (p. 2842), **decaf::util::StlList< Pointer< DestinationInfo > >** (p. 2842), **decaf::util::StlList< PrimitiveValueNode >** (p. 2842), **decaf::util::StlList< Pointer< Command > >** (p. 2842), and **decaf::util::StlList< Pointer< BackupTransport > >** (p. 2842).

6.399.3.10 `template<typename E> virtual E decaf::util::List< E >::remove (std::size_t index) throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IndexOutOfBoundsException)` [pure virtual]

Removes the element at the specified position in this list.

Shifts any subsequent elements to the left (subtracts one from their indices). Returns the element that was removed from the list.

Parameters

index - the index of the element to be removed

Returns

the element previously at the specified position

Exceptions

IndexOutOfBoundsException - if the index is greater than size

UnsupportedOperationException - If the collection is non-modifiable.

Implemented in **decaf::util::StlList< E >** (p. 2843), **decaf::util::StlList< CompositeTask * >** (p. 2843), **decaf::util::StlList< URI >** (p. 2843), **decaf::util::StlList< Pointer<**

DestinationInfo > > (p. 2843), **decaf::util::StlList< PrimitiveValueNode >** (p. 2843), **decaf::util::StlList< Pointer< Command > >** (p. 2843), and **decaf::util::StlList< Pointer< BackupTransport > >** (p. 2843).

```
6.399.3.11  template<typename E> virtual E decaf::util::List< E >::set
              ( std::size_t index, const E & element ) throw (
                decaf::lang::exceptions::IndexOutOfBoundsException ) [pure virtual]
```

Replaces the element at the specified position in this list with the specified element.

Parameters

index - index of the element to replace

element - element to be stored at the specified position

Returns

the element previously at the specified position

Exceptions

IndexOutOfBoundsException - if the index is greater than size

Implemented in **decaf::util::StlList< E >** (p. 2844), **decaf::util::StlList< CompositeTask * >** (p. 2844), **decaf::util::StlList< URI >** (p. 2844), **decaf::util::StlList< Pointer< DestinationInfo > >** (p. 2844), **decaf::util::StlList< PrimitiveValueNode >** (p. 2844), **decaf::util::StlList< Pointer< Command > >** (p. 2844), and **decaf::util::StlList< Pointer< BackupTransport > >** (p. 2844).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/List.h`

6.400 decaf::util::ListIterator< E > Class Template Reference

An iterator for lists that allows the programmer to traverse the list in either direction, modify the list during iteration, and obtain the iterator's current position in the list.

```
#include <src/main/decaf/util/ListIterator.h>
```

Inheritance diagram for **decaf::util::ListIterator< E >**:

Public Member Functions

- virtual **~ListIterator** ()
- virtual void **add** (const E &e)=0 throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IllegalArgumentException)

Inserts the specified element into the list (optional operation).

- virtual void **set** (const E &e)=0 throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException)
Replaces the last element returned by next or previous with the specified element (optional operation).
- virtual bool **hasPrevious** () const =0
Returns true if this list iterator has more elements when traversing the list in the reverse direction.
- virtual E **previous** ()=0
Returns the previous element in the list.
- virtual int **nextIndex** () const =0
Returns the index of the element that would be returned by a subsequent call to next.
- virtual int **previousIndex** () const =0
Returns the index of the element that would be returned by a subsequent call to previous.

6.400.1 Detailed Description

`template<typename E> class decaf::util::ListIterator< E >`

An iterator for lists that allows the programmer to traverse the list in either direction, modify the list during iteration, and obtain the iterator's current position in the list. Note that the **remove()** (p. 1718) and **set(Object)** methods are not defined in terms of the cursor position; they are defined to operate on the last element returned by a call to **next()** (p. 1717) or **previous()** (p. 1873).

6.400.2 Constructor & Destructor Documentation

6.400.2.1 `template<typename E > virtual decaf::util::ListIterator< E >::~ListIterator () [inline, virtual]`

6.400.3 Member Function Documentation

6.400.3.1 `template<typename E > virtual void decaf::util::ListIterator< E >::add (const E & e) throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IllegalArgumentException) [pure virtual]`

Inserts the specified element into the list (optional operation).

The element is inserted immediately before the next element that would be returned by next, if any, and after the next element that would be returned by previous, if any. (If the list contains no elements, the new element becomes the sole element on the list.) The new element is inserted before the implicit cursor: a subsequent call to next would be unaffected, and a subsequent call to previous would return the new element. (This call increases by one the value that would be returned by a call to nextIndex or previousIndex.)

Parameters

e - the element to insert.

Exceptions

UnsupportedOperationException - if the add method is not supported by this list iterator.

IllegalArgumentException - if some aspect of this element prevents it from being added to this list.

6.400.3.2 `template<typename E> virtual bool decaf::util::ListIterator< E >::hasPrevious () const [pure virtual]`

Returns true if this list iterator has more elements when traversing the list in the reverse direction.

(In other words, returns true if previous would return an element rather than throwing an exception.)

Returns

true if the list iterator has more elements when traversing the list in the reverse direction.

6.400.3.3 `template<typename E> virtual int decaf::util::ListIterator< E >::nextIndex () const [pure virtual]`

Returns the index of the element that would be returned by a subsequent call to next.

(Returns list size if the list iterator is at the end of the list.)

Returns

the index of the element that would be returned by a subsequent call to next, or list size if list iterator is at end of list.

6.400.3.4 `template<typename E> virtual E decaf::util::ListIterator< E >::previous () [pure virtual]`

Returns the previous element in the list.

This method may be called repeatedly to iterate through the list backwards, or intermixed with calls to next to go back and forth. (Note that alternating calls to next and previous will return the same element repeatedly.)

Returns

the previous element in the list.

Exceptions

NoSuchElementException - if the iteration has no previous element.

6.400.3.5 `template<typename E> virtual int decaf::util::ListIterator< E >::previousIndex () const [pure virtual]`

Returns the index of the element that would be returned by a subsequent call to previous.

(Returns -1 if the list iterator is at the beginning of the list.)

Returns

the index of the element that would be returned by a subsequent call to `previous`, or -1 if list iterator is at beginning of list.

6.400.3.6 `template<typename E > virtual void decaf::util::ListIterator<E >::set (const E & e) throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException) [pure virtual]`

Replaces the last element returned by `next` or `previous` with the specified element (optional operation).

This call can be made only if neither `ListIterator.remove` (p.1718) nor `ListIterator.add` (p.1872) have been called after the last call to `next` or `previous`.

Parameters

`e` - the element with which to replace the last element returned by `next` or `previous`.

Exceptions

UnsupportedOperationException - if the add method is not supported by this list iterator.

IllegalArgumentException - if some aspect of this element prevents it from being added to this list.

IllegalStateException - if neither `next` nor `previous` have been called, or `remove` or `add` have been called after the last call to `next` or `previous`.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/ListIterator.h`

6.401 activemq::commands::LocalTransactionId Class Reference

```
#include <src/main/activemq/commands/LocalTransactionId.h>
```

Inheritance diagram for `activemq::commands::LocalTransactionId`:

Public Types

- `typedef decaf::lang::PointerComparator< LocalTransactionId > COMPARATOR`

Public Member Functions

- `LocalTransactionId ()`
- `LocalTransactionId (const LocalTransactionId &other)`

- virtual `~LocalTransactionId ()`

- virtual unsigned char `getDataStructureType ()` const

Get the unique identifier that this object and its own Marshaler share.

- virtual `LocalTransactionId * cloneDataStructure ()` const

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void `copyDataStructure (const DataStructure *src)`

Copy the contents of the passed object into this object's members, overwriting any existing data.

- virtual std::string `toString ()` const

*Returns a string containing the information for this **DataStructure** (p. 1372) such as its type and value of its elements.*

- virtual bool `equals (const DataStructure *value)` const

*Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent.*

- virtual long long `getValue ()` const

- virtual void `setValue (long long value)`

- virtual const `Pointer< ConnectionId > &getConnectionId ()` const

- virtual `Pointer< ConnectionId > &getConnectionId ()`

- virtual void `setConnectionId (const Pointer< ConnectionId > &connectionId)`

- virtual int `compareTo (const LocalTransactionId &value)` const

- virtual bool `equals (const LocalTransactionId &value)` const

- virtual bool `operator== (const LocalTransactionId &value)` const

- virtual bool `operator< (const LocalTransactionId &value)` const

- `LocalTransactionId & operator= (const LocalTransactionId &other)`

Static Public Attributes

- static const unsigned char `ID_LOCALTRANSACTIONID = 111`

Protected Attributes

- long long `value`
- `Pointer< ConnectionId > connectionId`

6.401.1 Member Typedef Documentation

6.401.1.1 `typedef decaf::lang::PointerComparator<LocalTransactionId>
activemq::commands::LocalTransactionId::COMPARATOR`

6.401.2 Constructor & Destructor Documentation

6.401.2.1 `activemq::commands::LocalTransactionId::LocalTransactionId ()`

6.401.2.2 `activemq::commands::LocalTransactionId::LocalTransactionId (const
LocalTransactionId & other)`

6.401.2.3 `virtual activemq::commands::LocalTransactionId::~~LocalTransactionId ()
[virtual]`

6.401.3 Member Function Documentation

6.401.3.1 `virtual LocalTransactionId* ac-
tivemq::commands::LocalTransactionId::cloneDataStructure () const
[virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

6.401.3.2 `virtual int activemq::commands::LocalTransactionId::compareTo (const
LocalTransactionId & value) const [virtual]`

6.401.3.3 `virtual void activemq::commands::LocalTransactionId::copyDataStructure
(const DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

6.401.3.4 `virtual bool activemq::commands::LocalTransactionId::equals (const
DataStructure * value) const [virtual]`

Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

6.401.3.5 `virtual bool activemq::commands::LocalTransactionId::equals (const LocalTransactionId & value) const [virtual]`

6.401.3.6 `virtual const Pointer<ConnectionId>& activemq::commands::LocalTransactionId::getConnectionId () const [virtual]`

6.401.3.7 `virtual Pointer<ConnectionId>& activemq::commands::LocalTransactionId::getConnectionId () [virtual]`

6.401.3.8 `virtual unsigned char activemq::commands::LocalTransactionId::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataStructure** (p. 1372) type copy.

6.401.3.9 `virtual long long activemq::commands::LocalTransactionId::getValue () const [virtual]`

6.401.3.10 `virtual bool activemq::commands::LocalTransactionId::operator< (const LocalTransactionId & value) const [virtual]`

6.401.3.11 `LocalTransactionId& activemq::commands::LocalTransactionId::operator= (const LocalTransactionId & other)`

6.401.3.12 `virtual bool activemq::commands::LocalTransactionId::operator== (const LocalTransactionId & value) const [virtual]`

6.401.3.13 `virtual void activemq::commands::LocalTransactionId::setConnectionId (const Pointer< ConnectionId > & connectionId) [virtual]`

6.401.3.14 `virtual void activemq::commands::LocalTransactionId::setValue (long long value) [virtual]`

6.401.3.15 `virtual std::string activemq::commands::LocalTransactionId::toString () const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1372) such as its type and value of its elements.

Returns

formatted string useful for debugging.

6.401.4 Field Documentation

- 6.401.4.1 `Pointer<ConnectionId> activemq::commands::LocalTransactionId::connectionId`
[protected]
- 6.401.4.2 `const unsigned char activemq::commands::LocalTransactionId::ID - LOCALTRANSACTIONID = 111` [static]
- 6.401.4.3 `long long activemq::commands::LocalTransactionId::value` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/LocalTransactionId.h`

6.402 activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 1878).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/LocalTransactionIdMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller`:

Public Member Functions

- **LocalTransactionIdMarshaller** ()
- virtual `~LocalTransactionIdMarshaller` ()
- virtual `commands::DataStructure * createObject ()` const
Creates a new instance of this marshalable type.
- virtual unsigned char `getDataStructureType ()` const
Get the Data Structure Type that identifies this Marshaler.
- virtual void `tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- virtual int `tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.
- virtual void `tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write a object instance to data output stream.

- virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.402.1 Detailed Description

Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p.1878). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.402.2 Constructor & Destructor Documentation

6.402.2.1 activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller::LocalTransactionIdMarshaller () [inline]

6.402.2.2 virtual activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller::~~LocalTransactionIdMarshaller () [inline, virtual]

6.402.3 Member Function Documentation

6.402.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1331).

6.402.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1336).

6.402.3.3 virtual void activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller** (p. 3020).

6.402.3.4 virtual void activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller** (p. 3020).

6.402.3.5 virtual int activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller::tightMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller` (p. 3021).

```
6.402.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller` (p. 3021).

```
6.402.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller** (p. 3022).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/LocalTransactionIdMarshaller.h`

6.403 activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 1882).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/LocalTransactionIdMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller**:

Public Member Functions

- **LocalTransactionIdMarshaller** ()
- virtual **~LocalTransactionIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.403.1 Detailed Description

Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p.1882). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.403.2 Constructor & Destructor Documentation

6.403.2.1 activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller::LocalTransactionIdMarshaller () [inline]

6.403.2.2 virtual activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller::~LocalTransactionIdMarshaller () [inline, virtual]

6.403.3 Member Function Documentation

6.403.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1331).

6.403.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1336).

6.403.3.3 virtual void activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller** (p. 3034).

6.403.3.4 virtual void activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller** (p. 3035).

6.403.3.5 virtual int activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller::tightMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller` (p. 3035).

```
6.403.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller` (p. 3036).

```
6.403.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller** (p. 3036).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/LocalTransactionIdMarshaller.h`

6.404 activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 1886).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/LocalTransactionIdMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller**:

Public Member Functions

- **LocalTransactionIdMarshaller** ()
- virtual **~LocalTransactionIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)`

Write a object instance to data output stream.

6.404.1 Detailed Description

Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p.1886). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.404.2 Constructor & Destructor Documentation

6.404.2.1 `activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller::LocalTransactionIdMarshaller () [inline]`

6.404.2.2 `virtual activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller::~LocalTransactionIdMarshaller () [inline, virtual]`

6.404.3 Member Function Documentation

6.404.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1331).

6.404.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1336).

6.404.3.3 virtual void activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller** (p. 3027).

6.404.3.4 virtual void activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller** (p. 3027).

6.404.3.5 virtual int activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller::tightMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller` (p. 3028).

```
6.404.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller` (p. 3028).

```
6.404.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException*** if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller** (p. 3029).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**LocalTransactionIdMarshaller.h**

6.405 activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 1890).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/LocalTransactionIdMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller**:

Public Member Functions

- **LocalTransactionIdMarshaller** ()
- virtual **~LocalTransactionIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)`

Write a object instance to data output stream.

6.405.1 Detailed Description

Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p.1890). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.405.2 Constructor & Destructor Documentation

6.405.2.1 `activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller::LocalTransactionIdMarshaller () [inline]`

6.405.2.2 `virtual activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller::~LocalTransactionIdMarshaller () [inline, virtual]`

6.405.3 Member Function Documentation

6.405.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.405.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.405.3.3 virtual void activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller` (p. 3023).

6.405.3.4 virtual void activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller` (p. 3024).

6.405.3.5 virtual int activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller::tightMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller` (p. 3024).

```
6.405.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller` (p. 3025).

```
6.405.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller** (p. 3025).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/LocalTransactionIdMarshaller.h`

6.406 activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 1894).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/LocalTransactionIdMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller**:

Public Member Functions

- **LocalTransactionIdMarshaller** ()
- virtual **~LocalTransactionIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshall an object instance from the data input stream.

- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)`

Write a object instance to data output stream.

6.406.1 Detailed Description

Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p.1894). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.406.2 Constructor & Destructor Documentation

6.406.2.1 `activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller::LocalTransactionIdMarshaller () [inline]`

6.406.2.2 `virtual activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller::~LocalTransactionIdMarshaller () [inline, virtual]`

6.406.3 Member Function Documentation

6.406.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.406.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.406.3.3 virtual void activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller** (p. 3031).

6.406.3.4 virtual void activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller** (p. 3031).

6.406.3.5 virtual int activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller::tightMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller` (p. 3032).

```
6.406.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller` (p. 3032).

```
6.406.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller** (p. 3033).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/**LocalTransactionIdMarshaller.h**

6.407 decaf::util::concurrent::locks::Lock Class Reference

Lock (p. 1898) implementations provide more extensive locking operations than can be obtained using synchronized statements.

```
#include <src/main/decaf/util/concurrent/locks/Lock.h>
```

Inheritance diagram for decaf::util::concurrent::locks::Lock:

Public Member Functions

- virtual **~Lock** ()
- virtual void **lock** ()=0 throw (decaf::lang::exceptions::RuntimeException)
Acquires the lock.
- virtual void **lockInterruptibly** ()=0 throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::InterruptedException)
Acquires the lock unless the current thread is interrupted.
- virtual bool **tryLock** ()=0 throw (decaf::lang::exceptions::RuntimeException)
Acquires the lock only if it is free at the time of invocation.
- virtual bool **tryLock** (long long time, const **TimeUnit** &unit)=0 throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::InterruptedException)
Acquires the lock if it is free within the given waiting time and the current thread has not been interrupted.
- virtual void **unlock** ()=0 throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)
Releases the lock.
- virtual **Condition** * **newCondition** ()=0 throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::UnsupportedOperationException)
*Returns a new **Condition** (p. 1040) instance that is bound to this **Lock** (p. 1898) instance.*

6.407.1 Detailed Description

Lock (p. 1898) implementations provide more extensive locking operations than can be obtained using synchronized statements. They allow more flexible structuring, may have quite different properties, and may support multiple associated **Condition** (p. 1040) objects.

A lock is a tool for controlling access to a shared resource by multiple threads. Commonly, a lock provides exclusive access to a shared resource: only one thread at a time can acquire the lock and all access to the shared resource requires that the lock be acquired first. However, some locks may allow concurrent access to a shared resource, such as the read lock of a **ReadWriteLock** (p. 2522).

While the scoping mechanism for synchronized statements makes it much easier to program with monitor locks, and helps avoid many common programming errors involving locks, there are occasions where you need to work with locks in a more flexible way. For example, some algorithms for traversing concurrently accessed data structures require the use of "hand-over-hand" or "chain locking": you acquire the lock of node A, then node B, then release A and acquire C, then release B and acquire D and so on. Implementations of the **Lock** (p. 1898) interface enable the use of such techniques by allowing a lock to be acquired and released in different scopes, and allowing multiple locks to be acquired and released in any order.

With this increased flexibility comes additional responsibility. The absence of block-structured locking removes the automatic release of locks that occurs with synchronized statements. In most cases, the following idiom should be used:

```
Lock (p. 1898) l = ...; l.lock(); try { // access the resource protected by this lock } catch(...) { l.unlock(); }
```

When locking and unlocking occur in different scopes, care must be taken to ensure that all code that is executed while the lock is held is protected by try-catch ensure that the lock is released when necessary.

Lock (p. 1898) implementations provide additional functionality over the use of synchronized methods and statements by providing a non-blocking attempt to acquire a lock (**tryLock()** (p. 1902)), an attempt to acquire the lock that can be interrupted (**lockInterruptibly()** (p. 1900)), and an attempt to acquire the lock that can timeout (**tryLock(long, TimeUnit)**).

Note that **Lock** (p. 1898) instances are just normal objects and can themselves be used as the target in a synchronized statement.

The three forms of lock acquisition (interruptible, non-interruptible, and timed) may differ in their performance characteristics, ordering guarantees, or other implementation qualities. Further, the ability to interrupt the ongoing acquisition of a lock may not be available in a given **Lock** (p. 1898) class. Consequently, an implementation is not required to define exactly the same guarantees or semantics for all three forms of lock acquisition, nor is it required to support interruption of an ongoing lock acquisition. An implementation is required to clearly document the semantics and guarantees provided by each of the locking methods. It must also obey the interruption semantics as defined in this interface, to the extent that interruption of lock acquisition is supported: which is either totally, or only on method entry.

As interruption generally implies cancellation, and checks for interruption are often infrequent, an implementation can favor responding to an interrupt over normal method return. This is true even if it can be shown that the interrupt occurred after another action may have unblocked the thread. An implementation should document this behavior.

Since

1.0

6.407.2 Constructor & Destructor Documentation

6.407.2.1 `virtual decaf::util::concurrent::locks::Lock::~~Lock () [inline, virtual]`

6.407.3 Member Function Documentation

6.407.3.1 `virtual void decaf::util::concurrent::locks::Lock::lock () throw (decaf::lang::exceptions::RuntimeException) [pure virtual]`

Acquires the lock.

If the lock is not available then the current thread becomes disabled for thread scheduling purposes and lies dormant until the lock has been acquired.

Implementation Considerations

A **Lock** (p.1898) implementation may be able to detect erroneous use of the lock, such as an invocation that would cause deadlock, and may throw an exception in such circumstances. The circumstances and the exception type must be documented by that **Lock** (p. 1898) implementation.

Exceptions

RuntimeException if an error occurs while acquiring the lock.

Implemented in **decaf::util::concurrent::locks::ReentrantLock** (p.2529).

6.407.3.2 `virtual void decaf::util::concurrent::locks::Lock::lockInterruptibly () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::InterruptedException) [pure virtual]`

Acquires the lock unless the current thread is interrupted.

Acquires the lock if it is available and returns immediately.

If the lock is not available then the current thread becomes disabled for thread scheduling purposes and lies dormant until one of two things happens:

* The lock is acquired by the current thread; or * Some other thread interrupts the current thread, and interruption of lock acquisition is supported.

If the current thread:

* has its interrupted status set on entry to this method; or * is interrupted while acquiring the lock, and interruption of lock acquisition is supported,

then InterruptedException is thrown and the current thread's interrupted status is cleared.

Implementation Considerations

The ability to interrupt a lock acquisition in some implementations may not be possible, and if possible may be an expensive operation. The programmer should be aware that this may be the case. An implementation should document when this is the case.

An implementation can favor responding to an interrupt over normal method return.

A **Lock** (p.1898) implementation may be able to detect erroneous use of the lock, such as an invocation that would cause deadlock, and may throw an exception in such circumstances. The circumstances and the exception type must be documented by that **Lock** (p. 1898) implementation.

Exceptions

RuntimeException if an error occurs while acquiring the lock.

InterruptedException if the current thread is interrupted while acquiring the lock (and interruption of lock acquisition is supported).

Implemented in **decaf::util::concurrent::locks::ReentrantLock** (p. 2530).

6.407.3.3 **virtual Condition* decaf::util::concurrent::locks::Lock::newCondition**
() throw (decaf::lang::exceptions::RuntimeException,
decaf::lang::exceptions::UnsupportedOperationException) [pure
 virtual]

Returns a new **Condition** (p. 1040) instance that is bound to this **Lock** (p. 1898) instance.

Before waiting on the condition the lock must be held by the current thread. A call to **Condition.await()** (p. 1042) will atomically release the lock before waiting and re-acquire the lock before the wait returns.

Implementation Considerations

The exact operation of the **Condition** (p. 1040) instance depends on the **Lock** (p. 1898) implementation and must be documented by that implementation.

Returns

A new **Condition** (p. 1040) instance for this **Lock** (p. 1898) instance the caller must delete the returned **Condition** (p. 1040) object when done with it.

Exceptions

RuntimeException if an error occurs while creating the **Condition** (p. 1040).

UnsupportedOperationException if this **Lock** (p. 1898) implementation does not support conditions

Implemented in **decaf::util::concurrent::locks::ReentrantLock** (p. 2530).

6.407.3.4 **virtual bool decaf::util::concurrent::locks::Lock::tryLock**
(long long time, const TimeUnit & unit)
throw (decaf::lang::exceptions::RuntimeException,
decaf::lang::exceptions::InterruptedException) [pure virtual]

Acquires the lock if it is free within the given waiting time and the current thread has not been interrupted.

If the lock is available this method returns immediately with the value true. If the lock is not available then the current thread becomes disabled for thread scheduling purposes and lies dormant until one of three things happens:

* The lock is acquired by the current thread; or * Some other thread interrupts the current thread, and interruption of lock acquisition is supported; or * The specified waiting time elapses

If the lock is acquired then the value true is returned.

If the current thread:

* has its interrupted status set on entry to this method; or * is interrupted while acquiring the lock, and interruption of lock acquisition is supported,

then InterruptedException is thrown and the current thread's interrupted status is cleared.

If the specified waiting time elapses then the value false is returned. If the time is less than or equal to zero, the method will not wait at all.

Implementation Considerations

The ability to interrupt a lock acquisition in some implementations may not be possible, and if possible may be an expensive operation. The programmer should be aware that this may be the case. An implementation should document when this is the case.

An implementation can favor responding to an interrupt over normal method return, or reporting a timeout.

A **Lock** (p.1898) implementation may be able to detect erroneous use of the lock, such as an invocation that would cause deadlock, and may throw an (unchecked) exception in such circumstances. The circumstances and the exception type must be documented by that **Lock** (p.1898) implementation.

Parameters

time the maximum time to wait for the lock

unit the time unit of the time argument

Returns

true if the lock was acquired and false if the waiting time elapsed before the lock was acquired

Exceptions

RuntimeException if an error occurs while acquiring the lock.

InterruptedException if the current thread is interrupted while acquiring the lock (and interruption of lock acquisition is supported)

Implemented in **decaf::util::concurrent::locks::ReentrantLock** (p.2531).

6.407.3.5 virtual bool decaf::util::concurrent::locks::Lock::tryLock () throw (decaf::lang::exceptions::RuntimeException) [pure virtual]

Acquires the lock only if it is free at the time of invocation.

Acquires the lock if it is available and returns immediately with the value true. If the lock is not available then this method will return immediately with the value false.

A typical usage idiom for this method would be:

```
Lock (p.1898) lock = ...; if (lock.tryLock()) { try { // manipulate protected state } catch(...) { lock.unlock(); } } else { // perform alternative actions }
```

This usage ensures that the lock is unlocked if it was acquired, and doesn't try to unlock if the lock was not acquired.

Returns

true if the lock was acquired and false otherwise

Exceptions

RuntimeException if an error occurs while acquiring the lock.

Implemented in **decaf::util::concurrent::locks::ReentrantLock** (p. 2532).

6.407.3.6 virtual void decaf::util::concurrent::locks::Lock::unlock
() throw (decaf::lang::exceptions::RuntimeException,
decaf::lang::exceptions::IllegalMonitorStateException) [pure virtual]

Releases the lock.

Implementation Considerations

A **Lock** (p. 1898) implementation will usually impose restrictions on which thread can release a lock (typically only the holder of the lock can release it) and may throw an exception if the restriction is violated. Any restrictions and the exception type must be documented by that **Lock** (p. 1898) implementation.

Exceptions

RuntimeException if an error occurs while acquiring the lock.

IllegalMonitorStateException if the current thread is not the owner of the lock.

Implemented in **decaf::util::concurrent::locks::ReentrantLock** (p. 2533).

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/locks/**Lock.h**

6.408 decaf::util::concurrent::Lock Class Reference

A wrapper class around a given synchronization mechanism that provides automatic release upon destruction.

```
#include <src/main/decaf/util/concurrent/Lock.h>
```

Public Member Functions

- **Lock** (**Synchronizable** *object, const bool initiallyLocked=true)
Constructor - initializes the object member and locks the object if desired.
- virtual ~**Lock** ()
Destructor - Unlocks the object if it is locked.
- void **lock** ()
Locks the object.
- void **unlock** ()
Unlocks the object if it is already locked, otherwise a call to this method has no effect.
- bool **isLocked** () const
Indicates whether or not the object is locked.

6.408.1 Detailed Description

A wrapper class around a given synchronization mechanism that provides automatic release upon destruction.

Since

1.0

6.408.2 Constructor & Destructor Documentation

6.408.2.1 decaf::util::concurrent::Lock::Lock (Synchronizable * *object*, const bool *initiallyLocked* = *true*) [inline]

Constructor - initializes the object member and locks the object if desired.

Parameters

object The sync object to control

initiallyLocked If true, the object will automatically be locked.

References DECAF_CATCH_RETHROW, DECAF_CATCHALL_THROW, and lock().

6.408.2.2 virtual decaf::util::concurrent::Lock::~Lock () [inline, virtual]

Destructor - Unlocks the object if it is locked.

References DECAF_CATCH_RETHROW, DECAF_CATCHALL_THROW, and decaf::util::concurrent::Synchronizable::unlock().

6.408.3 Member Function Documentation

6.408.3.1 bool decaf::util::concurrent::Lock::isLocked () const [inline]

Indicates whether or not the object is locked.

Returns

true if the object is locked, otherwise false.

6.408.3.2 void decaf::util::concurrent::Lock::lock () [inline]

Locks the object.

References DECAF_CATCH_RETHROW, DECAF_CATCHALL_THROW, and decaf::util::concurrent::Synchronizable::lock().

Referenced by Lock().

6.408.3.3 void decaf::util::concurrent::Lock::unlock () [inline]

Unlocks the object if it is already locked, otherwise a call to this method has no effect.

References DECAF_CATCH_RETHROW, DECAF_CATCHALL_THROW, and decaf::util::concurrent::Synchronizable::unlock().

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**Lock.h**

6.409 decaf::util::concurrent::locks::LockSupport Class Reference

Basic thread blocking primitives for creating locks and other synchronization classes.

```
#include <src/main/decaf/util/concurrent/locks/LockSupport.h>
```

Public Member Functions

- ~LockSupport ()

Static Public Member Functions

- static void **unpark** (decaf::lang::Thread *thread) throw ()
Makes available the permit for the given thread, if it was not already available.
- static void **park** () throw ()
Disables the current thread for thread scheduling purposes unless the permit is available.
- static void **parkNanos** (long long nanos) throw ()
Disables the current thread for thread scheduling purposes, for up to the specified waiting time, unless the permit is available.
- static void **parkUntil** (long long deadline) throw ()
Disables the current thread for thread scheduling purposes, until the specified deadline, unless the permit is available.

6.409.1 Detailed Description

Basic thread blocking primitives for creating locks and other synchronization classes. This class associates, with each thread that uses it, a permit (in the sense of the **Semaphore** (p.2651) class). A call to park will return immediately if the permit is available, consuming it in the process; otherwise it may block. A call to unpark makes the permit available, if it was not already available. (Unlike with Semaphores though, permits do not accumulate. There is at most one.)

Methods park and unpark provide efficient means of blocking and unblocking threads. Races between one thread invoking park and another thread trying to unpark it will preserve liveness, due to the permit. Additionally, park will return if the caller's thread was interrupted, and timeout versions are supported. The park method may also return at any other time, for "no reason", so

in general must be invoked within a loop that rechecks conditions upon return. In this sense park serves as an optimization of a "busy wait" that does not waste as much time spinning, but must be paired with an unpark to be effective.

These methods are designed to be used as tools for creating higher-level synchronization utilities, and are not in themselves useful for most concurrency control applications. The park method is designed for use only in constructions of the form:

```
while (!canProceed()) { ... LockSupport.park(this); }
```

where neither canProceed nor any other actions prior to the call to park entail locking or blocking. Because only one permit is associated with each thread, any intermediary uses of park could interfere with its intended effects.

Sample Usage. Here is a sketch of a first-in-first-out non-reentrant lock class:

```
class FIFOMutex { private:
    AtomicBoolean locked; ConcurrentLinkedQueue<Thread*> waiters;
public:
    void lock() {
        bool wasInterrupted = false; Thread* current = Thread::currentThread(); waiters.add( current );
        // Block while not first in queue or cannot acquire lock while( waiters.peek() != current ||
        !locked.compareAndSet( false, true ) ) {
            LockSupport.park(this); if( Thread::interrupted() ) // ignore interrupts while waiting wasInter-
            rupted = true; }
        waiters.remove(); if( wasInterrupted ) // reassert interrupt status on exit current.interrupt(); }
    void unlock() { locked.set( false ); LockSupport.unpark (p.1907)( waiters.peek() ); } };
```

Since

1.0

6.409.2 Constructor & Destructor Documentation

6.409.2.1 decaf::util::concurrent::locks::LockSupport::~~LockSupport ()

6.409.3 Member Function Documentation

6.409.3.1 static void decaf::util::concurrent::locks::LockSupport::park () throw () [static]

Disables the current thread for thread scheduling purposes unless the permit is available.

If the permit is available then it is consumed and the call returns immediately; otherwise the current thread becomes disabled for thread scheduling purposes and lies dormant until one of three things happens:

- * Some other thread invokes unpark with the current thread as the target; or
- * Some other thread interrupts the current thread; or
- * The call spuriously (that is, for no reason) returns.

This method does not report which of these caused the method to return. Callers should re-check the conditions which caused the thread to park in the first place. Callers may also determine, for example, the interrupt status of the thread upon return.

6.409.3.2 `static void decaf::util::concurrent::locks::LockSupport::parkNanos (long long nanos) throw () [static]`

Disables the current thread for thread scheduling purposes, for up to the specified waiting time, unless the permit is available.

If the permit is available then it is consumed and the call returns immediately; otherwise the current thread becomes disabled for thread scheduling purposes and lies dormant until one of four things happens:

* Some other thread invokes `unpark` with the current thread as the target; or * Some other thread interrupts the current thread; or * The specified waiting time elapses; or * The call spuriously (that is, for no reason) returns.

This method does not report which of these caused the method to return. Callers should re-check the conditions which caused the thread to park in the first place. Callers may also determine, for example, the interrupt status of the thread, or the elapsed time upon return.

Parameters

nanos the maximum number of nanoseconds to wait

6.409.3.3 `static void decaf::util::concurrent::locks::LockSupport::parkUntil (long long deadline) throw () [static]`

Disables the current thread for thread scheduling purposes, until the specified deadline, unless the permit is available.

If the permit is available then it is consumed and the call returns immediately; otherwise the current thread becomes disabled for thread scheduling purposes and lies dormant until one of four things happens:

* Some other thread invokes `unpark` with the current thread as the target; or * Some other thread interrupts the current thread; or * The specified deadline passes; or * The call spuriously (that is, for no reason) returns.

This method does not report which of these caused the method to return. Callers should re-check the conditions which caused the thread to park in the first place. Callers may also determine, for example, the interrupt status of the thread, or the current time upon return.

Parameters

deadline the absolute time, in milliseconds from the Epoch, to wait until

6.409.3.4 `static void decaf::util::concurrent::locks::LockSupport::unpark (decaf::lang::Thread * thread) throw () [static]`

Makes available the permit for the given thread, if it was not already available.

If the thread was blocked on `park` then it will unblock. Otherwise, its next call to `park` is guaranteed not to block. This operation is not guaranteed to have any effect at all if the given thread has not been started.

Parameters

thread the thread to unport, or NULL in which case the method has no effect.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/locks/LockSupport.h`

6.410 decaf::util::logging::Logger Class Reference

```
#include <src/main/decaf/util/logging/Logger.h>
```

Public Member Functions

- **Logger** (const std::string &name, **Logger** *parent)
*Creates a new instance of the **Logger** (p. 1908) with the given name and assign it the given parent logger.*
- virtual ~**Logger** ()
- virtual const std::string & **getName** () const
*Gets the name of this **Logger** (p. 1908).*
- virtual void **addHandler** (**Handler** *handler) throw (lang::exceptions::IllegalArgumentException)
*Add a log **Handler** (p. 1604) to receive logging messages.*
- virtual void **removeHandler** (**Handler** *handler)
*Removes the specified **Handler** (p. 1604) and destroys it.*
- virtual void **setFilter** (**Filter** *filter)
Gets a vector containing all the handlers that this class has been assigned to use.
- virtual const **Filter** * **getFilter** () const
*Gets the **Filter** (p. 1527) object that this class is using.*
- virtual **Level** **getLevel** () const
*Get the log Level that has been specified for this **Logger** (p. 1908).*
- virtual void **setLevel** (**Level** level)
Set (p. 2729) the log level specifying which message levels will be logged by this logger.
- virtual bool **getUseParentHandlers** () const
Discover whether or not this logger is sending its output to its parent logger.
- virtual void **setUseParentHandlers** (bool value)
*pecify whether or not this logger should send its output to it's parent **Logger** (p. 1908).*
- virtual void **entry** (const std::string &blockName, const std::string &file, const int line)
Logs an Block Enter message.
- virtual void **exit** (const std::string &blockName, const std::string &file, const int line)
Logs an Block Exit message.

- virtual void **debug** (const std::string &file, const int line, const std::string functionName, const std::string &message)

Log a Debug Level Log.

- virtual void **info** (const std::string &file, const int line, const std::string functionName, const std::string &message)

Log a info Level Log.

- virtual void **warn** (const std::string &file, const int line, const std::string functionName, const std::string &message)

Log a warn Level Log.

- virtual void **error** (const std::string &file, const int line, const std::string functionName, const std::string &message)

Log a error Level Log.

- virtual void **fatal** (const std::string &file, const int line, const std::string functionName, const std::string &message)

Log a fatal Level Log.

- virtual bool **isLoggable** (**Level** level) const

Log a Throw Message.

- virtual void **log** (**LogRecord** &record)

*Log a **LogRecord** (p. 1928).*

- virtual void **log** (**Level** level, const std::string &message)

Log a message, with no arguments.

- virtual void **log** (**Level** level, const std::string &file, const int line, const std::string &message,...)

Log a message, with the list of params that is formatted into the message string.

- virtual void **log** (**Level** level, const std::string &file, const int line, const std::string &message, **lang::Exception** &ex)

Log a message, with associated Throwable information.

Static Public Member Functions

- static **Logger** * **getAnonymousLogger** ()

Creates an anonymous logger.

- static **Logger** * **getLogger** (const std::string &name)

Find or create a logger for a named subsystem.

6.410.1 Constructor & Destructor Documentation

6.410.1.1 decaf::util::logging::Logger::Logger (const std::string & *name*, Logger * *parent*)

Creates a new instance of the **Logger** (p. 1908) with the given name and assign it the given parent logger.

The logger will be initially configured with a null Level and with useParentHandlers true.

Parameters

name - A name for the logger. This should be a dot-separated name and should normally be based on the package name or class name of the subsystem, such as java.net or javax.swing. It may be null for anonymous Loggers.

parent logger that is this one's parent

6.410.1.2 virtual decaf::util::logging::Logger::~~Logger () [virtual]

6.410.2 Member Function Documentation

6.410.2.1 virtual void decaf::util::logging::Logger::addHandler (Handler * *handler*) throw (lang::exceptions::IllegalArgumentException) [virtual]

Add a log **Handler** (p. 1604) to receive logging messages.

By default, Loggers also send their output to their parent logger. Typically the root **Logger** (p. 1908) is configured with a set of Handlers that essentially act as default handlers for all loggers.

Parameters

handler A Logging **Handler** (p. 1604)

Exceptions

IllegalArgumentException

6.410.2.2 virtual void decaf::util::logging::Logger::debug (const std::string & *file*, const int *line*, const std::string *functionName*, const std::string & *message*) [virtual]

Log a Debug Level Log.

If the logger is currently enabled for the DEBUG message level then the given message is forwarded to all the registered output **Handler** (p. 1604) objects.

Parameters

file the file name where the log was generated

line the line number where the log was generated

functionName name of the function that logged this

message the message to log

6.410.2.3 `virtual void decaf::util::logging::Logger::entry (const std::string & blockName, const std::string & file, const int line) [virtual]`

Logs an Block Enter message.

This is a convenience method that is used to tag a block enter, a log record with the class name function name and the string Entering is logged at the DEBUG log level.

Parameters

blockName source block name

file source file name

line source line name

6.410.2.4 `virtual void decaf::util::logging::Logger::error (const std::string & file, const int line, const std::string fnctionName, const std::string & message) [virtual]`

Log a error Level Log.

If the logger is currently enabled for the error message level then the given message is forwarded to all the registered output **Handler** (p.1604) objects.

Parameters

file the file name where the log was generated

line the line number where the log was generated

fnctionName name of the function that logged this

message the message to log

6.410.2.5 `virtual void decaf::util::logging::Logger::exit (const std::string & blockName, const std::string & file, const int line) [virtual]`

Logs an Block Exit message.

This is a convenience method that is used to tag a block exit, a log record with the class name function name and the string Exiting is logged at the DEBUG log level.

Parameters

blockName source block name

file source file name

line source line name

6.410.2.6 `virtual void decaf::util::logging::Logger::fatal (const std::string & file, const int line, const std::string functionName, const std::string & message) [virtual]`

Log a fatal Level Log.

If the logger is currently enabled for the fatal message level then the given message is forwarded to all the registered output **Handler** (p.1604) objects.

Parameters

file the file name where the log was generated
line the line number where the log was generated
functionName name of the function that logged this
message the message to log

6.410.2.7 `static Logger* decaf::util::logging::Logger::getAnonymousLogger ()`
[static]

Creates an anonymous logger.

The newly created **Logger** (p. 1908) is not registered in the **LogManager** (p. 1923) namespace. There will be no access checks on updates to the logger. Even although the new logger is anonymous, it is configured to have the root logger ("") as its parent. This means that by default it inherits its effective level and handlers from the root logger.

The caller is responsible for destroying the returned logger.

Returns

Newly created anonymous logger

6.410.2.8 `virtual const Filter* decaf::util::logging::Logger::getFilter () const`
[inline, virtual]

Gets the **Filter** (p. 1527) object that this class is using.

Returns

the **Filter** (p. 1527) in use, can be null

6.410.2.9 `virtual Level decaf::util::logging::Logger::getLevel () const` [inline, virtual]

Get the log Level that has been specified for this **Logger** (p. 1908).

The result may be the Null level, which means that this logger's effective level will be inherited from its parent.

Returns

the level that is currently set

6.410.2.10 `static Logger* decaf::util::logging::Logger::getLogger (const std::string & name)` [static]

Find or create a logger for a named subsystem.

If a logger has already been created with the given name it is returned. Otherwise a new logger is created.

If a new logger is created its log level will be configured based on the **LogManager** (p. 1923) and it will be configured to also send logging output to its parent loggers Handlers. It will be registered in the **LogManager** (p. 1923) global namespace.

Parameters

name - A name for the logger. This should be a dot-separated name and should normally be based on the package name or class name of the subsystem, such as `cms` or `activemq.core.ActiveMQConnection` (p. 202)

Returns

a suitable logger.

6.410.2.11 `virtual const std::string& decaf::util::logging::Logger::getName ()`
`const [inline, virtual]`

Gets the name of this **Logger** (p. 1908).

Returns

logger name

6.410.2.12 `virtual bool decaf::util::logging::Logger::getUseParentHandlers ()`
`const [inline, virtual]`

Discover whether or not this logger is sending its output to its parent logger.

Returns

true if using Parent Handlers

6.410.2.13 `virtual void decaf::util::logging::Logger::info (const std::string & file,`
`const int line, const std::string functionName, const std::string &`
`message) [virtual]`

Log a info Level Log.

If the logger is currently enabled for the info message level then the given message is forwarded to all the registered output **Handler** (p. 1604) objects.

Parameters

file the file name where the log was generated
line the line number where the log was generated
functionName name of the function that logged this
message the message to log

6.410.2.14 `virtual bool decaf::util::logging::Logger::isLoggable (Level level)`
`const [virtual]`

Log a Throw Message.

If the logger is currently enabled for the Throwing message level then the given message is forwarded to all the registered output **Handler** (p. 1604) objects.

Parameters

file the file name where the log was generated

line the line number where the log was generated

functionName name of the function that logged this

message the message to log virtual void throwing(const std::string& file, const int line, const std::string functionName, const std::string& message); Check if a message of the given level would actually be logged by this logger. This check is based on the Loggers effective level, which may be inherited from its parent.

level - a message logging level

Returns

true if the given message level is currently being logged.

6.410.2.15 virtual void decaf::util::logging::Logger::log (Level *level*, const std::string & *message*) [virtual]

Log a message, with no arguments.

If the logger is currently enabled for the given message level then the given message is forwarded to all the registered output **Handler** (p.1604) objects

Parameters

level the Level to log at

message the message to log

6.410.2.16 virtual void decaf::util::logging::Logger::log (LogRecord & *record*) [virtual]

Log a **LogRecord** (p.1928).

All the other logging methods in this class call through this method to actually perform any logging. Subclasses can override this single method to capture all log activity.

Parameters

record - the **LogRecord** (p.1928) to be published

6.410.2.17 virtual void decaf::util::logging::Logger::log (Level *level*, const std::string & *file*, const int *line*, const std::string & *message*, ...) [virtual]

Log a message, with the list of params that is formatted into the message string.

If the logger is currently enabled for the given message level then the given message is forwarded to all the registered output **Handler** (p.1604) objects

Parameters

level the Level to log at

file the message to log

line the line in the file

... variable length argument to format the message string.

6.410.2.18 `virtual void decaf::util::logging::Logger::log (Level level, const std::string & file, const int line, const std::string & message, lang::Exception & ex)` [virtual]

Log a message, with associated Throwable information.

If the logger is currently enabled for the given message level then the given arguments are stored in a **LogRecord** (p. 1928) which is forwarded to all registered output handlers. Note that the thrown argument is stored in the **LogRecord** (p. 1928) thrown property, rather than the **LogRecord** (p. 1928) parameters property. Thus is it processed specially by output Formatters and is not treated as a formatting parameter to the **LogRecord** (p. 1928) message property.

Parameters

level the Level to log at.

file File that the message was logged in.

line the line number where the message was logged at.

message the message to log.

ex the Exception to log

6.410.2.19 `virtual void decaf::util::logging::Logger::removeHandler (Handler * handler)` [virtual]

Removes the specified **Handler** (p. 1604) and destroys it.

Returns silently if the given **Handler** (p. 1604) is not found.

Parameters

handler The **Handler** (p. 1604) to remove

6.410.2.20 `virtual void decaf::util::logging::Logger::setFilter (Filter * filter)` [virtual]

Gets a vector containing all the handlers that this class has been assigned to use.

Returns

a list of handlers that are used by this logger **Set** (p. 2729) a filter to control output on this **Logger** (p. 1908).

After passing the initial "level" check, the **Logger** (p. 1908) will call this **Filter** (p. 1527) to check if a log record should really be published.

The caller releases ownership of this filter to this logger

Parameters

filter to use, can be null

6.410.2.21 `virtual void decaf::util::logging::Logger::setLevel (Level level)`
`[inline, virtual]`

Set (p. 2729) the log level specifying which message levels will be logged by this logger.

Message levels lower than this value will be discarded. The level value `Level.OFF` can be used to turn off logging.

If the new level is the Null Level, it means that this node should inherit its level from its nearest ancestor with a specific (non-null) level value.

Parameters

level new Level value

6.410.2.22 `virtual void decaf::util::logging::Logger::setUseParentHandlers (bool value)`
`[inline, virtual]`

pecify whether or not this logger should send its output to it's parent **Logger** (p. 1908).

This means that any LogRecords will also be written to the parent's Handlers, and potentially to its parent, recursively up the namespace.

Parameters

value True is output is to be written to the parent

6.410.2.23 `virtual void decaf::util::logging::Logger::warn (const std::string & file,
const int line, const std::string functionName, const std::string &
message)` `[virtual]`

Log a warn Level Log.

If the logger is currently enabled for the warn message level then the given message is forwarded to all the registered output **Handler** (p. 1604) objects.

Parameters

file the file name where the log was generated

line the line number where the log was generated

functionName name of the function that logged this

message the message to log

The documentation for this class was generated from the following file:

- `src/main/decaf/util/logging/Logger.h`

6.411 decaf::util::logging::LoggerHierarchy Class Reference

```
#include <src/main/decaf/util/logging/LoggerHierarchy.h>
```

Public Member Functions

- **LoggerHierarchy** ()
- virtual **~LoggerHierarchy** ()

6.411.1 Constructor & Destructor Documentation

6.411.1.1 **decaf::util::logging::LoggerHierarchy::LoggerHierarchy** ()

6.411.1.2 **virtual decaf::util::logging::LoggerHierarchy::~~LoggerHierarchy** ()
[virtual]

The documentation for this class was generated from the following file:

- `src/main/decaf/util/logging/LoggerHierarchy.h`

6.412 activemq::io::LoggingInputStream Class Reference

```
#include <src/main/activemq/io/LoggingInputStream.h>
```

Inheritance diagram for `activemq::io::LoggingInputStream`:

Public Member Functions

- **LoggingInputStream** (`decaf::io::InputStream *inputStream`, `bool own=false`)
Creates a DataInputStream that uses the specified underlying InputStream.
- virtual **~LoggingInputStream** ()
- virtual `int read ()` throw (`decaf::io::IOException`)
Reads a single byte from the buffer.
- virtual `int read (unsigned char *buffer, std::size_t offset, std::size_t bufferSize)` throw (`decaf::io::IOException`, `decaf::lang::exceptions::NullPointerException`)
Reads an array of bytes from the buffer.

6.412.1 Constructor & Destructor Documentation

6.412.1.1 **activemq::io::LoggingInputStream::LoggingInputStream** (
`decaf::io::InputStream * inputStream`, `bool own = false`)

Creates a DataInputStream that uses the specified underlying InputStream.

Parameters

inputStream the InputStream instance to wrap.

own indicates if this class owns the wrapped string defaults to false.

6.412.1.2 virtual `activemq::io::LoggingInputStream::~~LoggingInputStream ()`
[virtual]

6.412.2 Member Function Documentation

6.412.2.1 virtual `int activemq::io::LoggingInputStream::read ()` throw (`decaf::io::IOException`) [virtual]

Reads a single byte from the buffer.

Blocks until data is available.

Returns

The next byte.

Exceptions

IOException thrown if an error occurs.

Reimplemented from `decaf::io::FilterInputStream` (p.1532).

6.412.2.2 virtual `int activemq::io::LoggingInputStream::read (unsigned char * buffer, std::size_t offset, std::size_t bufferSize)` throw (`decaf::io::IOException`, `decaf::lang::exceptions::NullPointerException`) [virtual]

Reads an array of bytes from the buffer.

Blocks until the requested number of bytes are available.

Parameters

buffer (out) the target buffer.

offset the position in the buffer to start at

bufferSize the size of the output buffer.

Returns

The number of bytes read or -1 if EOF is detected

Exceptions

IOException thrown if an error occurs.

NullPointerException if buffer is null

Reimplemented from `decaf::io::FilterInputStream` (p.1533).

The documentation for this class was generated from the following file:

- `src/main/activemq/io/LoggingInputStream.h`

6.413 activemq::io::LoggingOutputStream Class Reference

OutputStream filter that just logs the data being written.

```
#include <src/main/activemq/io/LoggingOutputStream.h>
```

Inheritance diagram for activemq::io::LoggingOutputStream:

Public Member Functions

- **LoggingOutputStream** (OutputStream *next, bool **own**=false)
Constructor.
- virtual **~LoggingOutputStream** ()
- virtual void **write** (unsigned char c) throw (decaf::io::IOException)
Writes a single byte to the output stream.
- virtual void **write** (const unsigned char *buffer, std::size_t offset, std::size_t len) throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException)
Writes an array of bytes to the output stream.

6.413.1 Detailed Description

OutputStream filter that just logs the data being written.

6.413.2 Constructor & Destructor Documentation

6.413.2.1 activemq::io::LoggingOutputStream::LoggingOutputStream (OutputStream * next, bool own = false)

Constructor.

Parameters

- next** The OutputStream to wrap an write logs to.
- own** If true, this object will control the lifetime of the output stream that it encapsulates.

6.413.2.2 virtual activemq::io::LoggingOutputStream::~~LoggingOutputStream () [virtual]

6.413.3 Member Function Documentation

6.413.3.1 virtual void activemq::io::LoggingOutputStream::write (unsigned char c) throw (decaf::io::IOException) [virtual]

Writes a single byte to the output stream.

Parameters

c the byte.

Exceptions

IOException thrown if an error occurs.

Reimplemented from `decaf::io::FilterOutputStream` (p. 1542).

6.413.3.2 `virtual void activemq::io::LoggingOutputStream::write (const unsigned char * buffer, std::size_t offset, std::size_t len) throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException)`
[virtual]

Writes an array of bytes to the output stream.

Parameters

buffer The array of bytes to write.

offset the position in the buffer to start at

len The number of bytes from the buffer to be written.

Exceptions

IOException thrown if an error occurs.

NullPointerException if buffer is null.

Reimplemented from `decaf::io::FilterOutputStream` (p. 1543).

The documentation for this class was generated from the following file:

- `src/main/activemq/io/LoggingOutputStream.h`

6.414 activemq::transport::logging::LoggingTransport Class Reference

A transport filter that logs commands as they are sent/received.

```
#include <src/main/activemq/transport/logging/LoggingTransport.h>
```

Inheritance diagram for `activemq::transport::logging::LoggingTransport`:

Public Member Functions

- `LoggingTransport (const Pointer< Transport > &next)`
Constructor.
- `virtual ~LoggingTransport ()`
- `virtual void onCommand (const Pointer< Command > &command)`

Event handler for the receipt of a command.

- virtual void **oneway** (const **Pointer**< **Command** > &command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)

Sends a one-way command.

- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)

Not supported by this class - throws an exception.

- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command, unsigned int timeout) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)

Not supported by this class - throws an exception.

6.414.1 Detailed Description

A transport filter that logs commands as they are sent/received.

6.414.2 Constructor & Destructor Documentation

6.414.2.1 activemq::transport::logging::LoggingTransport::LoggingTransport (const **Pointer**< **Transport** > & *next*)

Constructor.

Parameters

next - the next **Transport** (p. 3066) in the chain

6.414.2.2 virtual activemq::transport::logging::LoggingTransport::~~LoggingTransport () [inline, virtual]

6.414.3 Member Function Documentation

6.414.3.1 virtual void activemq::transport::logging::LoggingTransport::onCommand (const **Pointer**< **Command** > & *command*) [virtual]

Event handler for the receipt of a command.

Parameters

command - the received command object.

Reimplemented from **activemq::transport::TransportFilter** (p. 3077).

6.414.3.2 `virtual void activemq::transport::logging::LoggingTransport::oneway (const Pointer< Command > & command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Sends a one-way command.

Does not wait for any response from the broker.

Parameters

command the command to be sent.

Exceptions

IOException if an exception occurs during writing of the command.

UnsupportedOperationException if this method is not implemented by this transport.

Reimplemented from `activemq::transport::TransportFilter` (p. 3077).

6.414.3.3 `virtual Pointer<Response> activemq::transport::logging::LoggingTransport::request (const Pointer< Command > & command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Not supported by this class - throws an exception.

Parameters

command the command that is sent as a request

Exceptions

UnsupportedOperationException.

Reimplemented from `activemq::transport::TransportFilter` (p. 3079).

6.414.3.4 `virtual Pointer<Response> activemq::transport::logging::LoggingTransport::request (const Pointer< Command > & command, unsigned int timeout) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Not supported by this class - throws an exception.

Parameters

command the command that is sent as a request

timeout the time to wait for a response.

Exceptions

UnsupportedOperationException.

Reimplemented from `activemq::transport::TransportFilter` (p. 3078).

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/logging/LoggingTransport.h`

6.415 decaf::util::logging::LogManager Class Reference

There is a single global **LogManager** (p.1923) object that is used to maintain a set of shared state about Loggers and log services.

```
#include <src/main/decaf/util/logging/LogManager.h>
```

Public Member Functions

- virtual **~LogManager** ()
- virtual void **setProperties** (const **util::Properties** &properties)
*Sets the **Properties** (p.2494) this **LogManager** (p.1923) should use to configure its loggers.*
- virtual const **util::Properties** & **getProperties** () const
*Gets a reference to the Logging **Properties** (p.2494) used by this logger.*
- virtual std::string **getProperty** (const std::string &name)
*Gets the value of a named property of this **LogManager** (p.1923).*
- virtual void **addPropertyChangeListener** (PropertyChangeListener *listener)
*Adds a change listener for **LogManager** (p.1923) **Properties** (p.2494), adding the same instance of a change event listener does nothing.*
- virtual void **removePropertyChangeListener** (PropertyChangeListener *listener)
*Removes a properties change listener from the **LogManager** (p.1923).*
- virtual **Logger** * **getLogger** (const std::string &name)
*Retrieves or creates a new **Logger** (p.1908) using the name specified a new logger inherits the configuration of the logger's parent if there is no configuration data for the logger.*
- virtual int **getLoggerNames** (const std::vector< std::string > &names)
*Gets a list of known **Logger** (p.1908) Names from this Manager.*

Static Public Member Functions

- static **LogManager** * **getInstance** ()
Get the singleton instance.
- static void **returnInstance** ()
Returns a Checked out instance of this Manager.
- static void **destroy** ()
*Forcefully Delete the Instance of this **LogManager** (p.1923) even if there are outstanding references.*

Protected Member Functions

- **LogManager** ()
Constructor, hidden to protect against direct instantiation.
- **LogManager** (const **LogManager** &manager)
Copy Constructo.
- void **operator=** (const **LogManager** &manager)
Assignment operator.

6.415.1 Detailed Description

There is a single global **LogManager** (p.1923) object that is used to maintain a set of shared state about Loggers and log services. This **LogManager** (p.1923) object:

* Manages a hierarchical namespace of **Logger** (p.1908) objects. All named Loggers are stored in this namespace. * Manages a set of logging control properties. These are simple key-value pairs that can be used by Handlers and other logging objects to configure themselves.

The global **LogManager** (p.1923) object can be retrieved using `LogManager.getLogManager()`. The **LogManager** (p.1923) object is created during class initialization and cannot subsequently be changed.

TODO By default, the **LogManager** (p.1923) reads its initial configuration from a properties file "lib/logging.properties" in the JRE directory. If you edit that property file you can change the default logging configuration for all uses of that JRE.

In addition, the **LogManager** (p.1923) uses two optional system properties that allow more control over reading the initial configuration:

* "decaf.logger.config.class" * "decaf.logger.config.file"

These two properties may be set via the Preferences API, or as command line property definitions to the "java" command, or as system property definitions passed to `JNI_CreateJavaVM`.

If the "java.util.logging.config.class" property is set, then the property value is treated as a class name. The given class will be loaded, an object will be instantiated, and that object's constructor is responsible for reading in the initial configuration. (That object may use other system properties to control its configuration.) The alternate configuration class can use `readConfiguration(InputStream)` to define properties in the **LogManager** (p.1923).

If "java.util.logging.config.class" property is not set, then the "java.util.logging.config.file" system property can be used to specify a properties file (in `java.util.Properties` format). The initial logging configuration will be read from this file.

If neither of these properties is defined then, as described above, the **LogManager** (p.1923) will read its initial configuration from a properties file "lib/logging.properties" in the JRE directory.

The properties for loggers and Handlers will have names starting with the dot-separated name for the handler or logger. ***TODO***

The global logging properties may include:

* A property "handlers". This defines a whitespace separated list of class names for handler classes to load and register as handlers on the root **Logger** (p.1908) (the **Logger** (p.1908) named ""). Each class name must be for a **Handler** (p.1604) class which has a default constructor. Note that these Handlers may be created lazily, when they are first used. * A property "<logger>.handlers".

This defines a whitespace or comma separated list of class names for handlers classes to load and register as handlers to the specified logger. Each class name must be for a **Handler** (p. 1604) class which has a default constructor. Note that these Handlers may be created lazily, when they are first used. * A property "<logger>.useParentHandlers". This defines a boolean value. By default every logger calls its parent in addition to handling the logging message itself, this often result in messages being handled by the root logger as well. When setting this property to false a **Handler** (p. 1604) needs to be configured for this logger otherwise no logging messages are delivered. * A property "config". This property is intended to allow arbitrary configuration code to be run. The property defines a whitespace separated list of class names. A new instance will be created for each named class. The default constructor of each class may execute arbitrary code to update the logging configuration, such as setting logger levels, adding handlers, adding filters, etc.

Loggers are organized into a naming hierarchy based on their dot separated names. Thus "a.b.c" is a child of "a.b", but "a.b1" and "a.b2" are peers.

All properties whose names end with ".level" are assumed to define log levels for Loggers. Thus "foo.level" defines a log level for the logger called "foo" and (recursively) for any of its children in the naming hierarchy. Log Levels are applied in the order they are defined in the properties file. Thus level settings for child nodes in the tree should come after settings for their parents. The property name ".level" can be used to set the level for the root of the tree.

All methods on the **LogManager** (p. 1923) object are multi-thread safe.

6.415.2 Constructor & Destructor Documentation

6.415.2.1 `virtual decaf::util::logging::LogManager::~~LogManager () [virtual]`

6.415.2.2 `decaf::util::logging::LogManager::LogManager () [inline, protected]`

Constructor, hidden to protect against direct instantiation.

6.415.2.3 `decaf::util::logging::LogManager::LogManager (const LogManager & manager) [protected]`

Copy Constructo.

Parameters

manager the Manager to copy

6.415.3 Member Function Documentation

6.415.3.1 `virtual void decaf::util::logging::LogManager::addPropertyChangeListener (PropertyChangeListener * listener) [virtual]`

Adds a change listener for **LogManager** (p. 1923) **Properties** (p. 2494), adding the same instance of a change event listener does nothing.

Parameters

listener a PropertyChangeListener

6.415.3.2 static void decaf::util::logging::LogManager::destroy () [static]

Forcefully Delete the Instance of this **LogManager** (p.1923) even if there are outstanding references.

6.415.3.3 static LogManager* decaf::util::logging::LogManager::getInstance () [static]

Get the singleton instance.

Returns

Pointer to an instance of the Log Manager

6.415.3.4 virtual Logger* decaf::util::logging::LogManager::getLogger (const std::string & name) [virtual]

Retrieves or creates a new **Logger** (p.1908) using the name specified a new logger inherits the configuration of the logger's parent if there is no configuration data for the logger.

Parameters

name The name of the **Logger** (p.1908).

6.415.3.5 virtual int decaf::util::logging::LogManager::getLoggerNames (const std::vector< std::string > & names) [virtual]

Gets a list of known **Logger** (p.1908) Names from this Manager.

Parameters

names STL Vector to hold string logger names

Returns

names count of how many loggers were inserted

6.415.3.6 virtual const util::Properties& decaf::util::logging::LogManager::getProperties () const [inline, virtual]

Gets a reference to the Logging **Properties** (p.2494) used by this logger.

Returns

The **Logger** (p.1908) **Properties** (p.2494) Pointer

6.415.3.7 `virtual std::string decaf::util::logging::LogManager::getProperty (const std::string & name)` [inline, virtual]

Gets the value of a named property of this **LogManager** (p. 1923).

Parameters

name of the Property to retrieve

Returns

the value of the property

References decaf::util::Properties::getProperty().

6.415.3.8 `void decaf::util::logging::LogManager::operator= (const LogManager & manager)` [protected]

Assignment operator.

Parameters

manager the manager to assign from

6.415.3.9 `virtual void decaf::util::logging::LogManager::removePropertyChangeListener (PropertyChangeListener * listener)` [virtual]

Removes a properties change listener from the **LogManager** (p. 1923).

Parameters

listener a PropertyChangeListener

6.415.3.10 `static void decaf::util::logging::LogManager::returnInstance ()` [static]

Returns a Checked out instance of this Manager.

6.415.3.11 `virtual void decaf::util::logging::LogManager::setProperties (const util::Properties & properties)` [virtual]

Sets the **Properties** (p. 2494) this **LogManager** (p. 1923) should use to configure its loggers. Once set a properties change event is fired.

Parameters

properties Pointer to read the configuration from

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/**LogManager.h**

6.416 decaf::util::logging::LogRecord Class Reference

```
#include <src/main/decaf/util/logging/LogRecord.h>
```

Public Member Functions

- **LogRecord** ()
- virtual **~LogRecord** ()
- **Level** **getLevel** () const
Get Level of this log record.
- void **setLevel** (**Level** value)
Set (p. 2729) the Level of this Log Record.
- const std::string & **getLoggerName** () const
Gets the Source Logger's Name.
- void **setLoggerName** (const std::string &loggerName)
Sets the Source Logger's Name.
- const std::string & **getSourceFile** () const
Gets the Source Log File name.
- void **setSourceFile** (const std::string &sourceFile)
Sets the Source Log File Name.
- unsigned long **getSourceLine** () const
Gets the Source Log line number.
- void **setSourceLine** (long sourceLine)
Sets the Source Log line number.
- const std::string & **getMessage** () const
Gets the Message to be Logged.
- void **setMessage** (const std::string &message)
Sets the Message to be Logged.
- const std::string & **getSourceFunction** () const
Gets the name of the function where this log was logged.
- void **setSourceFunction** (const std::string &functionName)
Sets the name of the function where this log was logged.
- unsigned long **getTimestamp** () const
Gets the time in mills that this message was logged.
- void **setTimestamp** (long timeStamp)
Sets the time in mills that this message was logged.

- unsigned long **getThreadId** () const
Gets the Thread Id where this Log was created.
- void **setThreadId** (long threadId)
Sets the Thread Id where this Log was created.

6.416.1 Constructor & Destructor Documentation

6.416.1.1 **decaf::util::logging::LogRecord::LogRecord** () [inline]

6.416.1.2 **virtual decaf::util::logging::LogRecord::~~LogRecord** () [inline, virtual]

6.416.2 Member Function Documentation

6.416.2.1 **Level decaf::util::logging::LogRecord::getLevel** () const [inline]

Get Level of this log record.

Returns

Level enumeration value.

6.416.2.2 **const std::string& decaf::util::logging::LogRecord::getLoggerName** () const [inline]

Gets the Source Logger's Name.

Returns

the source loggers name

6.416.2.3 **const std::string& decaf::util::logging::LogRecord::getMessage** () const [inline]

Gets the Message to be Logged.

Returns

the source logger's message

Referenced by decaf::util::logging::SimpleFormatter::formatMessage().

6.416.2.4 **const std::string& decaf::util::logging::LogRecord::getSourceFile** () const [inline]

Gets the Source Log File name.

Returns

the source loggers name

6.416.2.5 `const std::string& decaf::util::logging::LogRecord::getSourceFunction () const [inline]`

Gets the name of the function where this log was logged.

Returns

the source logger's message

6.416.2.6 `unsigned long decaf::util::logging::LogRecord::getSourceLine () const [inline]`

Gets the Source Log line number.

Returns

the source loggers line number

6.416.2.7 `unsigned long decaf::util::logging::LogRecord::getTimestamp () const [inline]`

Gets the time in mills that this message was logged.

Returns

UTC time in milliseconds

6.416.2.8 `unsigned long decaf::util::logging::LogRecord::getTreadId () const [inline]`

Gets the Thread Id where this Log was created.

Returns

the source loggers line number

6.416.2.9 `void decaf::util::logging::LogRecord::setLevel (Level value) [inline]`

Set (p. 2729) the Level of this Log Record.

Parameters

value Level Enumeration Value

6.416.2.10 `void decaf::util::logging::LogRecord::setLoggerName (const std::string & loggerName) [inline]`

Sets the Source Logger's Name.

Parameters

loggerName the source loggers name

6.416.2.11 `void decaf::util::logging::LogRecord::setMessage (const std::string & message) [inline]`

Sets the Message to be Logged.

Parameters

message the source loggers message

6.416.2.12 `void decaf::util::logging::LogRecord::setSourceFile (const std::string & sourceFile) [inline]`

Sets the Source Log File Name.

Parameters

sourceFile the source loggers name

6.416.2.13 `void decaf::util::logging::LogRecord::setSourceFunction (const std::string & functionName) [inline]`

Sets the name of the function where this log was logged.

Parameters

functionName the source of the log

6.416.2.14 `void decaf::util::logging::LogRecord::setSourceLine (long sourceLine) [inline]`

Sets the Source Log line number.

Parameters

sourceLine the source logger's line number

6.416.2.15 `void decaf::util::logging::LogRecord::setTimestamp (long timeStamp) [inline]`

Sets the time in mills that this message was logged.

Parameters

timeStamp UTC Time in Milliseconds.

6.416.2.16 void decaf::util::logging::LogRecord::setTreadId (long *threadId*) [inline]

Sets the Thread Id where this Log was created.

Parameters

- threadId* the source logger's line number

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/**LogRecord.h**

6.417 decaf::util::logging::LogWriter Class Reference

```
#include <src/main/decaf/util/logging/LogWriter.h>
```

Public Member Functions

- **LogWriter** ()
- virtual **~LogWriter** ()
- virtual void **log** (const std::string &file, const int line, const std::string &prefix, const std::string &message)
Writes a message to the output destination.
- virtual void **log** (const std::string &message)
Writes a message to the output destination.

Static Public Member Functions

- static **LogWriter** & **getInstance** ()
Get the singleton instance.
- static void **returnInstance** ()
Returns a Checked out instance of this Writer.
- static void **destroy** ()
*Forcefully Delete the Instance of this **LogWriter** (p. 1932) even if there are outstanding references.*

6.417.1 Constructor & Destructor Documentation

6.417.1.1 `decaf::util::logging::LogWriter::LogWriter ()`

6.417.1.2 `virtual decaf::util::logging::LogWriter::~~LogWriter ()` [virtual]

6.417.2 Member Function Documentation

6.417.2.1 `static void decaf::util::logging::LogWriter::destroy ()` [static]

Forcefully Delete the Instance of this **LogWriter** (p. 1932) even if there are outstanding references.

6.417.2.2 `static LogWriter& decaf::util::logging::LogWriter::getInstance ()`
[static]

Get the singleton instance.

6.417.2.3 `virtual void decaf::util::logging::LogWriter::log (const std::string & message)` [virtual]

Writes a message to the output destination.

Parameters

message

6.417.2.4 `virtual void decaf::util::logging::LogWriter::log (const std::string & file,
const int line, const std::string & prefix, const std::string & message
)` [virtual]

Writes a message to the output destination.

Parameters

file

line

prefix

message

6.417.2.5 `static void decaf::util::logging::LogWriter::returnInstance ()` [static]

Returns a Checked out instance of this Writer.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/logging/LogWriter.h`

6.418 decaf::lang::Long Class Reference

#include <src/main/decaf/lang/Long.h>

Inheritance diagram for decaf::lang::Long:

Public Member Functions

- **Long** (long long value)
- **Long** (const std::string &value) throw (exceptions::NumberFormatException)
- virtual ~**Long** ()
- virtual int **compareTo** (const **Long** &l) const
*Compares this **Long** (p. 1934) instance with another.*
- bool **equals** (const **Long** &l) const
- virtual bool **operator==** (const **Long** &l) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const **Long** &l) const
Compares this object to another and returns true if this object is considered to be less than the one passed.
- virtual int **compareTo** (const long long &l) const
*Compares this **Long** (p. 1934) instance with another.*
- bool **equals** (const long long &l) const
- virtual bool **operator==** (const long long &l) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const long long &l) const
Compares this object to another and returns true if this object is considered to be less than the one passed.
- std::string **toString** () const
- virtual double **doubleValue** () const
Answers the double value which the receiver represents.
- virtual float **floatValue** () const
Answers the float value which the receiver represents.
- virtual unsigned char **byteValue** () const
Answers the byte value which the receiver represents.
- virtual short **shortValue** () const
Answers the short value which the receiver represents.
- virtual int **intValue** () const
Answers the int value which the receiver represents.

- virtual long long **longValue** () const
Answers the long value which the receiver represents.

Static Public Member Functions

- static int **bitCount** (long long value)
Returns the number of one-bits in the two's complement binary representation of the specified int value.
- static **Long decode** (const std::string &value) throw (exceptions::NumberFormatException)
*Decodes a String into a **Long** (p. 1934).*
- static long long **highestOneBit** (long long value)
Returns an long long value with at most a single one-bit, in the position of the highest-order ("leftmost") one-bit in the specified int value.
- static long long **lowestOneBit** (long long value)
Returns an long long value with at most a single one-bit, in the position of the lowest-order ("rightmost") one-bit in the specified int value.
- static int **numberOfLeadingZeros** (long long value)
Returns the number of zero bits preceding the highest-order ("leftmost") one-bit in the two's complement binary representation of the specified long long value.
- static int **numberOfTrailingZeros** (long long value)
Returns the number of zero bits following the lowest-order ("rightmost") one-bit in the two's complement binary representation of the specified long long value.
- static long long **parseLong** (const std::string &value) throw (exceptions::NumberFormatException)
Parses the string argument as a signed decimal long.
- static long long **parseLong** (const std::string &value, int radix) throw (exceptions::NumberFormatException)
*Returns a **Long** (p. 1934) object holding the value extracted from the specified string when parsed with the radix given by the second argument.*
- static long long **reverseBytes** (long long value)
Returns the value obtained by reversing the order of the bytes in the two's complement representation of the specified long long value.
- static long long **reverse** (long long value)
Returns the value obtained by reversing the order of the bits in the two's complement binary representation of the specified long long value.
- static long long **rotateLeft** (long long value, int distance)
Returns the value obtained by rotating the two's complement binary representation of the specified value left by the specified number of bits.

- static long long **rotateRight** (long long value, int distance)
Returns the value obtained by rotating the two's complement binary representation of the specified value right by the specified number of bits.
- static int **signum** (long long value)
Returns the signum function of the specified value.
- static std::string **toString** (long long value)
Converts the long to a String representation.
- static std::string **toString** (long long value, int radix)
- static std::string **toHexString** (long long value)
Returns a string representation of the integer argument as an unsigned integer in base 16.
- static std::string **toOctalString** (long long value)
Returns a string representation of the long long argument as an unsigned long long in base 8.
- static std::string **toBinaryString** (long long value)
Returns a string representation of the long long argument as an unsigned long long in base 2.
- static **Long** **valueOf** (long long value)
*Returns a **Long** (p. 1934) instance representing the specified int value.*
- static **Long** **valueOf** (const std::string &value) throw (exceptions::NumberFormatException)
*Returns a **Long** (p. 1934) object holding the value given by the specified std::string.*
- static **Long** **valueOf** (const std::string &value, int radix) throw (exceptions::NumberFormatException)
*Returns a **Long** (p. 1934) object holding the value extracted from the specified std::string when parsed with the radix given by the second argument.*

Static Public Attributes

- static const int **SIZE** = 64
The size in bits of the primitive long long type.
- static const long long **MAX_VALUE** = (long long)0x7FFFFFFFFFFFFFFFFLL
The maximum value that the primitive type can hold.
- static const long long **MIN_VALUE** = (long long)0x800000000000000LL
The minimum value that the primitive type can hold.

6.418.1 Constructor & Destructor Documentation

6.418.1.1 `decaf::lang::Long::Long (long long value)`

Parameters

value - the primitive long long to wrap

6.418.1.2 `decaf::lang::Long::Long (const std::string & value) throw (exceptions::NumberFormatException)`

Parameters

value - the long long formatted string to wrap

Exceptions

NumberFormatException

6.418.1.3 `virtual decaf::lang::Long::~~Long () [inline, virtual]`

6.418.2 Member Function Documentation

6.418.2.1 `static int decaf::lang::Long::bitCount (long long value) [static]`

Returns the number of one-bits in the two's complement binary representation of the specified int value.

This function is sometimes referred to as the population count.

Parameters

value - the long long to count

Returns

the number of one-bits in the two's complement binary representation of the specified long long value.

6.418.2.2 `virtual unsigned char decaf::lang::Long::byteValue () const [inline, virtual]`

Answers the byte value which the receiver represents.

Returns

int the value of the receiver.

6.418.2.3 `virtual int decaf::lang::Long::compareTo (const long long & l) const [virtual]`

Compares this **Long** (p. 1934) instance with another.

Parameters

l - the **Integer** (p. 1652) instance to be compared

Returns

zero if this object represents the same integer value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

Implements **decaf::lang::Comparable**< **long long** > (p. 1010).

6.418.2.4 virtual int decaf::lang::Long::compareTo (const Long & *l*) const
[virtual]

Compares this **Long** (p. 1934) instance with another.

Parameters

l - the **Long** (p. 1934) instance to be compared

Returns

zero if this object represents the same integer value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

6.418.2.5 static Long decaf::lang::Long::decode (const std::string & *value*)
throw (exceptions::NumberFormatException) [static]

Decodes a String into a **Long** (p. 1934).

Accepts decimal, hexadecimal, and octal numbers given by the following grammar:

The sequence of characters following an (optional) negative sign and/or radix specifier ("0x", "0X", "#", or leading zero) is parsed as by the Integer.parseInt method with the indicated radix (10, 16, or 8). This sequence of characters must represent a positive value or a NumberFormatException will be thrown. The result is negated if first character of the specified String is the minus sign. No whitespace characters are permitted in the string.

Parameters

value - The string to decode

Returns

a **Long** (p. 1934) object containing the decoded value

Exceptions

NumberFormatException if the string is not formatted correctly.

6.418.2.6 `virtual double decaf::lang::Long::doubleValue () const [inline, virtual]`

Answers the double value which the receiver represents.

Returns

double the value of the receiver.

6.418.2.7 `bool decaf::lang::Long::equals (const Long & l) const [inline]`

Parameters

l - the **Long** (p. 1934) object to compare against.

Returns

true if the two **Integer** (p. 1652) Objects have the same value.

6.418.2.8 `bool decaf::lang::Long::equals (const long long & l) const [inline, virtual]`

Parameters

l - the **Long** (p. 1934) object to compare against.

Returns

true if the two **Integer** (p. 1652) Objects have the same value.

Implements **decaf::lang::Comparable< long long >** (p. 1010).

6.418.2.9 `virtual float decaf::lang::Long::floatValue () const [inline, virtual]`

Answers the float value which the receiver represents.

Returns

float the value of the receiver.

6.418.2.10 `static long long decaf::lang::Long::highestOneBit (long long value) [static]`

Returns an long long value with at most a single one-bit, in the position of the highest-order ("leftmost") one-bit in the specified int value.

Returns zero if the specified value has no one-bits in its two's complement binary representation, that is, if it is equal to zero.

Parameters

value - the long long to be inspected

Returns

an long long value with a single one-bit, in the position of the highest-order one-bit in the specified value, or zero if the specified value is itself equal to zero.

6.418.2.11 virtual int decaf::lang::Long::intValue () const [inline, virtual]

Answers the int value which the receiver represents.

Returns

int the value of the receiver.

6.418.2.12 virtual long long decaf::lang::Long::longValue () const [inline, virtual]

Answers the long value which the receiver represents.

Returns

long the value of the receiver.

6.418.2.13 static long long decaf::lang::Long::lowestOneBit (long long value) [static]

Returns an long long value with at most a single one-bit, in the position of the lowest-order ("rightmost") one-bit in the specified int value.

Returns zero if the specified value has no one-bits in its two's complement binary representation, that is, if it is equal to zero.

Parameters

value - the long long to be inspected

Returns

an long long value with a single one-bit, in the position of the lowest-order one-bit in the specified value, or zero if the specified value is itself equal to zero.

6.418.2.14 static int decaf::lang::Long::numberOfLeadingZeros (long long value) [static]

Returns the number of zero bits preceding the highest-order ("leftmost") one-bit in the two's complement binary representation of the specified long long value.

Returns 64 if the specified value has no one-bits in its two's complement representation, in other words if it is equal to zero.

Note that this method is closely related to the logarithm base 2. For all positive int values x:

* floor(log2(x)) = 63 - numberOfLeadingZeros(x) * ceil(log2(x)) = 64 - numberOfLeadingZeros(x - 1)

Parameters

value - the long long to be inspected

Returns

the number of zero bits preceding the highest-order ("leftmost") one-bit in the two's complement binary representation of the specified long long value, or 64 if the value is equal to zero.

6.418.2.15 `static int decaf::lang::Long::numberOfTrailingZeros (long long value)`
[static]

Returns the number of zero bits following the lowest-order ("rightmost") one-bit in the two's complement binary representation of the specified long long value.

Returns 64 if the specified value has no one-bits in its two's complement representation, in other words if it is equal to zero.

Parameters

value - the int to be inspected

Returns

the number of zero bits following the lowest-order ("rightmost") one-bit in the two's complement binary representation of the specified long long value, or 64 if the value is equal to zero.

6.418.2.16 `virtual bool decaf::lang::Long::operator< (const long long & l) const`
[inline, virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

Parameters

l - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

Implements `decaf::lang::Comparable< long long >` (p. 1011).

6.418.2.17 `virtual bool decaf::lang::Long::operator< (const Long & l) const`
[inline, virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

Parameters

l - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

6.418.2.18 `virtual bool decaf::lang::Long::operator==(const Long & l) const`
[inline, virtual]

Compares equality between this object and the one passed.

Parameters

l - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

6.418.2.19 `virtual bool decaf::lang::Long::operator==(const long long & l)`
`const` [inline, virtual]

Compares equality between this object and the one passed.

Parameters

l - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

Implements `decaf::lang::Comparable< long long >` (p.1011).

6.418.2.20 `static long long decaf::lang::Long::parseLong (const std::string &`
`value) throw (exceptions::NumberFormatException)` [static]

Parses the string argument as a signed decimal long.

The characters in the string must all be decimal digits, except that the first character may be an ASCII minus sign '-' to indicate a negative value. The resulting long value is returned, exactly as if the argument and the radix 10 were given as arguments to the `parseLong(java.lang.String, int)` method.

Note that the characters LL or ULL are not permitted to appear at the end of this string as would normally be permitted in a C++ program.

Parameters

value - String to parse

Returns

long long value

Exceptions

NumberFormatException on invalid string value

6.418.2.21 `static long long decaf::lang::Long::parseLong (const std::string &
 value, int radix) throw (exceptions::NumberFormatException)
 [static]`

Returns a **Long** (p. 1934) object holding the value extracted from the specified string when parsed with the radix given by the second argument.

The first argument is interpreted as representing a signed long in the radix specified by the second argument, exactly as if the arguments were given to the `parseLong(std::string, int)` method. The result is a **Long** (p. 1934) object that represents the long long value specified by the string.

Parameters

value - String to parse

radix - the base encoding of the string

Returns

long long value

Exceptions

NumberFormatException on invalid string value

6.418.2.22 `static long long decaf::lang::Long::reverse (long long value) [static]`

Returns the value obtained by reversing the order of the bits in the two's complement binary representation of the specified long long value.

Parameters

value - the value whose bits are to be reversed

Returns

the reversed bits long long.

6.418.2.23 `static long long decaf::lang::Long::reverseBytes (long long value)
 [static]`

Returns the value obtained by reversing the order of the bytes in the two's complement representation of the specified long long value.

Parameters

value - the long long whose bytes we are to reverse

Returns

the reversed long long.

6.418.2.24 static long long decaf::lang::Long::rotateLeft (long long *value*, int *distance*) [static]

Returns the value obtained by rotating the two's complement binary representation of the specified value left by the specified number of bits.

(Bits shifted out of the left hand, or high-order, side reenter on the right, or low-order.)

Note that left rotation with a negative distance is equivalent to right rotation: rotateLeft(val, -distance) == rotateRight(val, distance). Note also that rotation by any multiple of 32 is a no-op, so all but the last five bits of the rotation distance can be ignored, even if the distance is negative: rotateLeft(val, distance) == rotateLeft(val, distance & 0x1F).

Parameters

value - the long long to be inspected

distance - the number of bits to rotate

Returns

the value obtained by rotating the two's complement binary representation of the specified value left by the specified number of bits.

6.418.2.25 static long long decaf::lang::Long::rotateRight (long long *value*, int *distance*) [static]

Returns the value obtained by rotating the two's complement binary representation of the specified value right by the specified number of bits.

(Bits shifted out of the right hand, or low-order, side reenter on the left, or high-order.)

Note that right rotation with a negative distance is equivalent to left rotation: rotateRight(val, -distance) == rotateLeft(val, distance). Note also that rotation by any multiple of 32 is a no-op, so all but the last five bits of the rotation distance can be ignored, even if the distance is negative: rotateRight(val, distance) == rotateRight(val, distance & 0x1F).

Parameters

value - the long long to be inspected

distance - the number of bits to rotate

Returns

the value obtained by rotating the two's complement binary representation of the specified value right by the specified number of bits.

6.418.2.26 virtual short decaf::lang::Long::shortValue () const [inline, virtual]

Answers the short value which the receiver represents.

Returns

int the value of the receiver.

6.418.2.27 static int decaf::lang::Long::signum (long long *value*) [static]

Returns the signum function of the specified value.

(The return value is -1 if the specified value is negative; 0 if the specified value is zero; and 1 if the specified value is positive.)

Parameters

value - the long long to be inspected

Returns

the signum function of the specified long long value.

6.418.2.28 static std::string decaf::lang::Long::toBinaryString (long long *value*) [static]

Returns a string representation of the long long argument as an unsigned long long in base 2.

The unsigned long long value is the argument plus 2^{32} if the argument is negative; otherwise it is equal to the argument. This value is converted to a string of ASCII digits in binary (base 2) with no extra leading 0s. If the unsigned magnitude is zero, it is represented by a single zero character '0'; otherwise, the first character of the representation of the unsigned magnitude will not be the zero character. The characters '0' and '1' are used as binary digits.

Parameters

value - the long long to be translated to a binary string

Returns

the unsigned long long value as a binary string

6.418.2.29 static std::string decaf::lang::Long::toHexString (long long *value*) [static]

Returns a string representation of the integer argument as an unsigned integer in base 16.

The unsigned integer value is the argument plus 2^{32} if the argument is negative; otherwise, it is equal to the argument. This value is converted to a string of ASCII digits in hexadecimal (base 16) with no extra leading 0s. If the unsigned magnitude is zero, it is represented by a single zero character '0'; otherwise, the first character of the representation of the unsigned magnitude will not be the zero character. The following characters are used as hexadecimal digits:

0123456789abcdef

If uppercase letters are desired, the toUpperCase() method may be called on the result:

Parameters

value - the long long to be translated to an Octal string

Returns

the unsigned long long value as a Octal string

6.418.2.30 `static std::string decaf::lang::Long::toOctalString (long long value)`
 [static]

Returns a string representation of the long long argument as an unsigned long long in base 8.

The unsigned long long value is the argument plus 2^{32} if the argument is negative; otherwise, it is equal to the argument. This value is converted to a string of ASCII digits in octal (base 8) with no extra leading 0s.

If the unsigned magnitude is zero, it is represented by a single zero character '0'; otherwise, the first character of the representation of the unsigned magnitude will not be the zero character. The following characters are used as octal digits:

01234567

Parameters

value - the long long to be translated to an Octal string

Returns

the unsigned long long value as a Octal string

6.418.2.31 `static std::string decaf::lang::Long::toString (long long value)`
 [static]

Converts the long to a String representation.

Parameters

value The long to convert to a std::string.

Returns

string representation

6.418.2.32 `std::string decaf::lang::Long::toString () const`

Returns

this **Long** (p. 1934) Object as a String Representation

6.418.2.33 `static std::string decaf::lang::Long::toString (long long value, int`
 `radix) [static]`

6.418.2.34 `static Long decaf::lang::Long::valueOf (long long value) [inline,`
 `static]`

Returns a **Long** (p. 1934) instance representing the specified int value.

Parameters

value - the long long to wrap

Returns

the new **Integer** (p. 1652) object wrapping value.

6.418.2.35 `static Long decaf::lang::Long::valueOf (const std::string & value, int radix) throw (exceptions::NumberFormatException) [static]`

Returns a **Long** (p.1934) object holding the value extracted from the specified std::string when parsed with the radix given by the second argument.

The first argument is interpreted as representing a signed long long in the radix specified by the second argument, exactly as if the argument were given to the `parseLong(std::string, int)` method. The result is a **Long** (p.1934) object that represents the long long value specified by the string.

Parameters

value - std::string to parse as base (radix)
radix - base of the string to parse.

Returns

new **Long** (p.1934) Object wrapping the primitive

Exceptions

NumberFormatException if the string is not a valid long long.

6.418.2.36 `static Long decaf::lang::Long::valueOf (const std::string & value) throw (exceptions::NumberFormatException) [static]`

Returns a **Long** (p.1934) object holding the value given by the specified std::string.

The argument is interpreted as representing a signed decimal long long, exactly as if the argument were given to the `parseLong(std::string)` method. The result is a **Integer** (p.1652) object that represents the long long value specified by the string.

Parameters

value - std::string to parse as base 10

Returns

new **Long** (p.1934) Object wrapping the primitive

Exceptions

NumberFormatException if the string is not a decimal long long.

6.418.3 Field Documentation

6.418.3.1 `const long long decaf::lang::Long::MAX_VALUE = (long long)0x7FFFFFFFFFFFFFFFFFLL [static]`

The maximum value that the primitive type can hold.

6.418.3.2 `const long long decaf::lang::Long::MIN_VALUE = (long long)0x8000000000000000LL [static]`

The minimum value that the primitive type can hold.

6.418.3.3 const int decaf::lang::Long::SIZE = 64 [static]

The size in bits of the primitive long long type.

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**Long.h**

6.419 decaf::internal::nio::LongArrayBuffer Class Reference

```
#include <src/main/decaf/internal/nio/LongArrayBuffer.h>
```

Inheritance diagram for decaf::internal::nio::LongArrayBuffer:

Public Member Functions

- **LongArrayBuffer** (std::size_t capacity, bool readOnly=false)
*Creates a **IntArrayBuffer** (p. 1634) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*
- **LongArrayBuffer** (long long *array, std::size_t offset, std::size_t capacity, bool readOnly=false) throw (decaf::lang::exceptions::NullPointerException)
*Creates a **LongArrayBuffer** (p. 1948) object that wraps the given array.*
- **LongArrayBuffer** (ByteArrayPerspective &array, std::size_t offset, std::size_t capacity, bool readOnly=false) throw (decaf::lang::exceptions::IndexOutOfBoundsException)
*Creates a byte buffer that wraps the passed **ByteArrayPerspective** (p. 843) and start at the given offset.*
- **LongArrayBuffer** (const LongArrayBuffer &other)
*Create a **LongArrayBuffer** (p. 1948) that mirrors this one, meaning it shares a reference to this buffers **ByteArrayPerspective** (p. 843) and when changes are made to that data it is reflected in both.*
- virtual ~**LongArrayBuffer** ()
- virtual long long * **array** () throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException)
Returns the long long array that backs this buffer (optional operation).
- virtual std::size_t **arrayOffset** () throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException)
Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).
- virtual LongBuffer * **asReadOnlyBuffer** () const
Creates a new, read-only long long buffer that shares this buffer's content.
- virtual LongBuffer & **compact** () throw (decaf::nio::ReadOnlyBufferException)
Compacts this buffer.

- virtual LongBuffer * **duplicate** ()
Creates a new long long buffer that shares this buffer's content.
- virtual long long **get** () throw (decaf::nio::BufferUnderflowException)
Relative get method.
- virtual long long **get** (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)
Absolute get method.
- virtual bool **hasArray** () const
Tells whether or not this buffer is backed by an accessible long long array.
- virtual bool **isReadOnly** () const
Tells whether or not this buffer is read-only.
- virtual LongBuffer & **put** (long long value) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException)
Writes the given doubles into this buffer at the current position, and then increments the position.
- virtual LongBuffer & **put** (std::size_t index, long long value) throw (lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException)
Writes the given doubles into this buffer at the given index.
- virtual LongBuffer * **slice** () const
Creates a new LongBuffer whose content is a shared subsequence of this buffer's content.

Protected Member Functions

- virtual void **setReadOnly** (bool value)
*Sets this **ByteArrayBuffer** (p. 806) as Read-Only.*

6.419.1 Constructor & Destructor Documentation

6.419.1.1 decaf::internal::nio::LongArrayBuffer::LongArrayBuffer (std::size_t capacity, bool readOnly = false)

Creates a **IntArrayBuffer** (p. 1634) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.

The array is initially created with all elements initialized to zero.

Parameters

capacity - size of the array, this is the limit we read and write to.

readOnly - should this buffer be read-only, default as false

6.419.1.2 `decaf::internal::nio::LongArrayBuffer::LongArrayBuffer (long long * array, std::size_t offset, std::size_t capacity, bool readOnly = false) throw (decaf::lang::exceptions::NullPointerException)`

Creates a **LongArrayBuffer** (p.1948) object that wraps the given array.

If the own flag is set then it will delete this array when this object is deleted.

Parameters

array - array to wrap

offset - the position that is this buffers start pos.

capacity - size of the array, this is the limit we read and write to.

readOnly - should this buffer be read-only, default as false

Exceptions

NullPointerException if buffer is NULL

6.419.1.3 `decaf::internal::nio::LongArrayBuffer::LongArrayBuffer (ByteArrayPerspective & array, std::size_t offset, std::size_t capacity, bool readOnly = false) throw (decaf::lang::exceptions::IndexOutOfBoundsException)`

Creates a byte buffer that wraps the passed **ByteArrayPerspective** (p.843) and start at the given offset.

The capacity and limit of the new **LongArrayBuffer** (p.1948) will be that of the remaining capacity of the passed buffer.

Parameters

array - the **ByteArrayPerspective** (p.843) to wrap

offset - the offset into array where the buffer starts

capacity - the length of the array we are wrapping or limit.

readOnly - is this a readOnly buffer.

Exceptions

IndexOutOfBoundsException if offset is greater than array capacity.

6.419.1.4 `decaf::internal::nio::LongArrayBuffer::LongArrayBuffer (const LongArrayBuffer & other)`

Create a **LongArrayBuffer** (p.1948) that mirrors this one, meaning it shares a reference to this buffers **ByteArrayPerspective** (p.843) and when changes are made to that data it is reflected in both.

Parameters

other - the **LongArrayBuffer** (p.1948) this one is to mirror.

6.419.1.5 `virtual decaf::internal::nio::LongArrayBuffer::~~LongArrayBuffer ()`
[virtual]

6.419.2 Member Function Documentation

6.419.2.1 `virtual long long* decaf::internal::nio::LongArrayBuffer::array ()`
`throw (decaf::lang::exceptions::UnsupportedOperationException,`
`decaf::nio::ReadOnlyBufferException)` [virtual]

Returns the long long array that backs this buffer (optional operation).

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

the array that backs this Buffer

Exceptions

ReadOnlyBufferException if this Buffer is read only.

UnsupportedOperationException if the underlying store has no array.

Implements `decaf::nio::LongBuffer` (p. 1958).

6.419.2.2 `virtual std::size_t decaf::internal::nio::LongArrayBuffer::arrayOffset (`
`) throw (decaf::lang::exceptions::UnsupportedOperationException,`
`decaf::nio::ReadOnlyBufferException)` [virtual]

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

The offset into the backing array where index zero starts.

Exceptions

ReadOnlyBufferException if this Buffer is read only.

UnsupportedOperationException if the underlying store has no array.

Implements `decaf::nio::LongBuffer` (p. 1959).

6.419.2.3 `virtual LongBuffer* de-`
`caf::internal::nio::LongArrayBuffer::asReadOnlyBuffer (`
`) const` [virtual]

Creates a new, read-only long long buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns

The new, read-only long long buffer which the caller then owns.

Implements **decaf::nio::LongBuffer** (p. 1959).

6.419.2.4 virtual LongBuffer& decaf::internal::nio::LongArrayBuffer::compact () throw (decaf::nio::ReadOnlyBufferException) [virtual]

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index $p = \text{position}()$ (p. 746) is copied to index zero, the byte at index $p + 1$ is copied to index one, and so forth until the byte at index $\text{limit}()$ (p. 745) - 1 is copied to index $n = \text{limit}()$ (p. 745) - 1 - p . The buffer's position is then set to $n+1$ and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns

a reference to this LongBuffer

Exceptions

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::LongBuffer** (p. 1959).

6.419.2.5 virtual LongBuffer* decaf::internal::nio::LongArrayBuffer::duplicate () [virtual]

Creates a new long long buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

a new long long Buffer which the caller owns.

Implements **decaf::nio::LongBuffer** (p. 1960).

6.419.2.6 `virtual long long decaf::internal::nio::LongArrayBuffer::get () throw (decaf::nio::BufferUnderflowException) [virtual]`

Relative get method.

Reads the value at this buffer's current position, and then increments the position.

Returns

the long long at the current position

Exceptions

BufferUnderflowException if there no more data to return

Implements `decaf::nio::LongBuffer` (p. 1962).

6.419.2.7 `virtual long long decaf::internal::nio::LongArrayBuffer::get (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException) [virtual]`

Absolute get method.

Reads the value at the given index.

Parameters

index - the index in the Buffer where the long long is to be read

Returns

the long long that is located at the given index

Exceptions

IndexOutOfBoundsException - If index is not smaller than the buffer's limit

Implements `decaf::nio::LongBuffer` (p. 1961).

6.419.2.8 `virtual bool decaf::internal::nio::LongArrayBuffer::hasArray () const [inline, virtual]`

Tells whether or not this buffer is backed by an accessible long long array.

If this method returns true then the `array` and `arrayOffset` methods may safely be invoked. Sub-classes should override this method if they do not have a backing array as this class always returns true.

Returns

true if, and only if, this buffer is backed by an array and is not read-only

Implements `decaf::nio::LongBuffer` (p. 1962).

6.419.2.9 `virtual bool decaf::internal::nio::LongArrayBuffer::isReadOnly () const`
[inline, virtual]

Tells whether or not this buffer is read-only.

Returns

true if, and only if, this buffer is read-only

6.419.2.10 `virtual LongBuffer& decaf::internal::nio::LongArrayBuffer::put`
`(std::size_t index, long long value) throw`
`(lang::exceptions::IndexOutOfBoundsException,`
`decaf::nio::ReadOnlyBufferException) [virtual]`

Writes the given doubles into this buffer at the given index.

Parameters

index - position in the Buffer to write the data

value - the doubles to write.

Returns

a reference to this buffer

Exceptions

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException - If this buffer is read-only

Implements `decaf::nio::LongBuffer` (p. 1963).

6.419.2.11 `virtual LongBuffer& decaf::internal::nio::LongArrayBuffer::put`
`(long long value) throw (decaf::nio::BufferOverflowException,`
`decaf::nio::ReadOnlyBufferException) [virtual]`

Writes the given doubles into this buffer at the current position, and then increments the position.

Parameters

value - the doubles value to be written

Returns

a reference to this buffer

Exceptions

BufferOverflowException - If this buffer's current position is not smaller than its limit

ReadOnlyBufferException - If this buffer is read-only

Implements `decaf::nio::LongBuffer` (p. 1963).

6.419.2.12 `virtual void decaf::internal::nio::LongArrayBuffer::setReadOnly (bool value)` [inline, protected, virtual]

Sets this **ByteBuffer** (p. 806) as Read-Only.

Parameters

value - true if this buffer is to be read-only.

6.419.2.13 `virtual LongBuffer* decaf::internal::nio::LongArrayBuffer::slice ()`
`const` [virtual]

Creates a new LongBuffer whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the newly create LongBuffer which the caller owns.

Implements **decaf::nio::LongBuffer** (p. 1965).

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/nio/LongArrayBuffer.h`

6.420 decaf::nio::LongBuffer Class Reference

This class defines four categories of operations upon long long buffers:

```
#include <src/main/decaf/nio/LongBuffer.h>
```

Inheritance diagram for decaf::nio::LongBuffer:

Public Member Functions

- `virtual ~LongBuffer ()`
- `virtual std::string toString () const`
- `virtual long long * array ()=0 throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException)`
Returns the long long array that backs this buffer (optional operation).
- `virtual std::size_t arrayOffset ()=0 throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException)`

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

- virtual **LongBuffer** * **asReadOnlyBuffer** () const =0
Creates a new, read-only long long buffer that shares this buffer's content.
- virtual **LongBuffer** & **compact** ()=0 throw (ReadOnlyBufferException)
Compacts this buffer.
- virtual **LongBuffer** * **duplicate** ()=0
Creates a new long long buffer that shares this buffer's content.
- virtual long long **get** ()=0 throw (BufferUnderflowException)
Relative get method.
- virtual long long **get** (std::size_t index) const =0 throw (lang::exceptions::IndexOutOfBoundsException)
Absolute get method.
- **LongBuffer** & **get** (std::vector< long long > buffer) throw (BufferUnderflowException)
Relative bulk get method.
- **LongBuffer** & **get** (long long *buffer, std::size_t offset, std::size_t length) throw (BufferUnderflowException, lang::exceptions::NullPointerException)
Relative bulk get method.
- virtual bool **hasArray** () const =0
Tells whether or not this buffer is backed by an accessible long long array.
- **LongBuffer** & **put** (**LongBuffer** &src) throw (BufferOverflowException, ReadOnlyBufferException, lang::exceptions::IllegalArgumentException)
This method transfers the long longs remaining in the given source buffer long longo this buffer.
- **LongBuffer** & **put** (const long long *buffer, std::size_t offset, std::size_t length) throw (BufferOverflowException, ReadOnlyBufferException, lang::exceptions::NullPointerException)
This method transfers long longs long longo this buffer from the given source array.
- **LongBuffer** & **put** (std::vector< long long > &buffer) throw (BufferOverflowException, ReadOnlyBufferException)
This method transfers the entire content of the given source long longs array long longo this buffer.
- virtual **LongBuffer** & **put** (long long value)=0 throw (BufferOverflowException, ReadOnlyBufferException)
Writes the given long longs long longo this buffer at the current position, and then increments the position.
- virtual **LongBuffer** & **put** (std::size_t index, long long value)=0 throw (lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException)
Writes the given long longs long longo this buffer at the given index.

- virtual **LongBuffer** * **slice** () const =0
*Creates a new **LongBuffer** (p. 1955) whose content is a shared subsequence of this buffer's content.*
- virtual int **compareTo** (const **LongBuffer** &value) const
Compares this object with the specified object for order.
- virtual bool **equals** (const **LongBuffer** &value) const
- virtual bool **operator==** (const **LongBuffer** &value) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const **LongBuffer** &value) const
Compares this object to another and returns true if this object is considered to be less than the one passed.

Static Public Member Functions

- static **LongBuffer** * **allocate** (std::size_t capacity)
Allocates a new Double buffer.
- static **LongBuffer** * **wrap** (long long *array, std::size_t offset, std::size_t length) throw (lang::exceptions::NullPointerException)
*Wraps the passed buffer with a new **LongBuffer** (p. 1955).*
- static **LongBuffer** * **wrap** (std::vector< long long > &buffer)
*Wraps the passed STL long long Vector in a **LongBuffer** (p. 1955).*

Protected Member Functions

- **LongBuffer** (std::size_t capacity)
*Creates a **LongBuffer** (p. 1955) object that has its backing array allocated long longernally and is then owned and deleted when this object is deleted.*

6.420.1 Detailed Description

This class defines four categories of operations upon long long buffers: o Absolute and relative get and put methods that read and write single long longs; o Relative bulk get methods that transfer contiguous sequences of long longs from this buffer long longo an array; and o Relative bulk put methods that transfer contiguous sequences of long longs from a long long array or some other long long buffer long longo this buffer o Methods for compacting, duplicating, and slicing a long long buffer.

Double buffers can be created either by allocation, which allocates space for the buffer's content, by wrapping an existing long long array long longo a buffer, or by creating a view of an existing byte buffer

Methods in this class that do not otherwise have a value to return are specified to return the buffer upon which they are invoked. This allows method invocations to be chained.

6.420.2 Constructor & Destructor Documentation

6.420.2.1 decaf::nio::LongBuffer::LongBuffer (std::size_t *capacity*) [protected]

Creates a **LongBuffer** (p. 1955) object that has its backing array allocated long longernally and is then owned and deleted when this object is deleted.

The array is initially created with all elements initialized to zero.

Parameters

capacity - size and limit of the **Buffer** (p. 741) in doubles

6.420.2.2 virtual decaf::nio::LongBuffer::~~LongBuffer () [inline, virtual]

6.420.3 Member Function Documentation

6.420.3.1 static LongBuffer* decaf::nio::LongBuffer::allocate (std::size_t *capacity*) [static]

Allocates a new Double buffer.

The new buffer's position will be zero, its limit will be its capacity, and its mark will be undefined. It will have a backing array, and its array offset will be zero.

Parameters

capacity - The size of the Double buffer in long longs

Returns

the **LongBuffer** (p. 1955) that was allocated, caller owns.

6.420.3.2 virtual long long* decaf::nio::LongBuffer::array () throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException) [pure virtual]

Returns the long long array that backs this buffer (optional operation).

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

the array that backs this **Buffer** (p. 741)

Exceptions

ReadOnlyBufferException (p. 2520) if this **Buffer** (p. 741) is read only.

UnsupportedOperationException if the underlying store has no array.

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 1951).

6.420.3.3 `virtual std::size_t decaf::nio::LongBuffer::arrayOffset ()
throw (decaf::lang::exceptions::UnsupportedOperationException,
ReadOnlyBufferException) [pure virtual]`

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

The offset long along the backing array where index zero starts.

Exceptions

ReadOnlyBufferException (p. 2520) if this **Buffer** (p. 741) is read only.

UnsupportedOperationException if the underlying store has no array.

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 1951).

6.420.3.4 `virtual LongBuffer* decaf::nio::LongBuffer::asReadOnlyBuffer () const
[pure virtual]`

Creates a new, read-only long long buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the `duplicate` method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns

The new, read-only long long buffer which the caller then owns.

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 1951).

6.420.3.5 `virtual LongBuffer& decaf::nio::LongBuffer::compact () throw (
ReadOnlyBufferException) [pure virtual]`

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index `p = position()` (p. 746) is copied to index zero, the byte at index `p + 1` is copied to index one, and so forth until the byte at index `limit()` (p. 745) - 1 is copied to index `n = limit()` (p. 745) - 1 - `p`. The buffer's position is then set to `n+1` and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns

a reference to this **LongBuffer** (p. 1955)

Exceptions

ReadOnlyBufferException (p. 2520) - If this buffer is read-only

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 1952).

6.420.3.6 virtual int decaf::nio::LongBuffer::compareTo (const LongBuffer & value) const [virtual]

Compares this object with the specified object for order.

Returns a negative long longeger, zero, or a positive long longeger as this object is less than, equal to, or greater than the specified object.

Parameters

value - the Object to be compared.

Returns

a negative long longeger, zero, or a positive long longeger as this object is less than, equal to, or greater than the specified object.

6.420.3.7 virtual LongBuffer* decaf::nio::LongBuffer::duplicate () [pure virtual]

Creates a new long long buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

a new long long **Buffer** (p. 741) which the caller owns.

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 1952).

6.420.3.8 virtual bool decaf::nio::LongBuffer::equals (const LongBuffer & value) const [virtual]**Returns**

true if this value is considered equal to the passed value.

6.420.3.9 LongBuffer& decaf::nio::LongBuffer::get (std::vector< long long > *buffer*) throw (BufferUnderflowException)

Relative bulk get method.

This method transfers values from this buffer long longo the given destination vector. An invocation of this method of the form `src.get(a)` behaves in exactly the same way as the invocation. The vector must be sized to the amount of data that is to be read, that is to say, the caller should call `buffer.resize(N)` before calling this get method.

Returns

a reference to this **Buffer** (p. 741)

Exceptions

BufferUnderflowException (p. 774) - If there are fewer than length long longs remaining in this buffer

6.420.3.10 virtual long long decaf::nio::LongBuffer::get (std::size_t *index*) const throw (lang::exceptions::IndexOutOfBoundsException) [pure virtual]

Absolute get method.

Reads the value at the given index.

Parameters

index - the index in the **Buffer** (p. 741) where the long long is to be read

Returns

the long long that is located at the given index

Exceptions

IndexOutOfBoundsException - If index is not smaller than the buffer's limit

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 1953).

6.420.3.11 LongBuffer& decaf::nio::LongBuffer::get (long long * *buffer*, std::size_t *offset*, std::size_t *length*) throw (BufferUnderflowException, lang::exceptions::NullPointerException)

Relative bulk get method.

This method transfers long longs from this buffer long longo the given destination array. If there are fewer long longs remaining in the buffer than are required to satisfy the request, that is, if `length > remaining()` (p. 746), then no bytes are transferred and a **BufferUnderflowException** (p. 774) is thrown.

Otherwise, this method copies length long longs from this buffer long longo the given array, starting at the current position of this buffer and at the given offset in the array. The position of this buffer is then incremented by length.

Parameters

buffer - long longer to an allocated buffer to fill
offset - position in the buffer to start filling
length - amount of data to put in the passed buffer

Returns

a reference to this **Buffer** (p. 741)

Exceptions

BufferUnderflowException (p. 774) - If there are fewer than length long longs remaining in this buffer
NullPointerException longerException if the passed buffer is null.

6.420.3.12 `virtual long long decaf::nio::LongBuffer::get () throw (BufferUnderflowException) [pure virtual]`

Relative get method.

Reads the value at this buffer's current position, and then increments the position.

Returns

the long long at the current position

Exceptions

BufferUnderflowException (p. 774) if there no more data to return

Implemented in **decaf::internal::nio::LongArrayBuffer** (p.1953).

6.420.3.13 `virtual bool decaf::nio::LongBuffer::hasArray () const [pure virtual]`

Tells whether or not this buffer is backed by an accessible long long array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Sub-classes should override this method if they do not have a backing array as this class always returns true.

Returns

true if, and only if, this buffer is backed by an array and is not read-only

Implemented in **decaf::internal::nio::LongArrayBuffer** (p.1953).

6.420.3.14 `virtual bool decaf::nio::LongBuffer::operator< (const LongBuffer & value) const [virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

Parameters

value - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

6.420.3.15 `virtual bool decaf::nio::LongBuffer::operator==(const LongBuffer & value) const` [virtual]

Compares equality between this object and the one passed.

Parameters

value - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

6.420.3.16 `virtual LongBuffer& decaf::nio::LongBuffer::put (long long value) throw (BufferOverflowException, ReadOnlyBufferException)` [pure virtual]

Writes the given long longs long longo this buffer at the current position, and then increments the position.

Parameters

value - the long longs value to be written

Returns

a reference to this buffer

Exceptions

BufferOverflowException (p. 771) - If this buffer's current position is not smaller than its limit

ReadOnlyBufferException (p. 2520) - If this buffer is read-only

Implemented in `decaf::internal::nio::LongArrayBuffer` (p.1954).

6.420.3.17 `virtual LongBuffer& decaf::nio::LongBuffer::put (std::size_t index, long long value) throw (lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException)` [pure virtual]

Writes the given long longs long longo this buffer at the given index.

Parameters

index - position in the **Buffer** (p. 741) to write the data

value - the long longs to write.

Returns

a reference to this buffer

Exceptions

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException (p. 2520) - If this buffer is read-only

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 1954).

6.420.3.18 **LongBuffer& decaf::nio::LongBuffer::put (const long long * *buffer*, std::size_t *offset*, std::size_t *length*) throw (BufferOverflowException, ReadOnlyBufferException, lang::exceptions::NullPointerException)**

This method transfers long longs long longo this buffer from the given source array.

If there are more long longs to be copied from the array than remain in this buffer, that is, if `length > remaining()` (p. 746), then no long longs are transferred and a **BufferOverflowException** (p. 771) is thrown.

Otherwise, this method copies `length` bytes from the given array long longo this buffer, starting at the given offset in the array and at the current position of this buffer. The position of this buffer is then incremented by `length`.

Parameters

buffer- The array from which long longs are to be read

offset- The offset within the array of the first long long to be read;

length - The number of long longs to be read from the given array

Returns

a reference to this buffer

Exceptions

BufferOverflowException (p. 771) - If there is insufficient space in this buffer

ReadOnlyBufferException (p. 2520) - If this buffer is read-only

NullPointerException if the passed buffer is null.

6.420.3.19 **LongBuffer& decaf::nio::LongBuffer::put (std::vector< long long > & *buffer*) throw (BufferOverflowException, ReadOnlyBufferException)**

This method transfers the entire content of the given source long longs array long longo this buffer.

This is the same as calling `put(&buffer[0], 0, buffer.size()`

Parameters

buffer - The buffer whose contents are copied to this **LongBuffer** (p. 1955)

Returns

a reference to this buffer

Exceptions

BufferOverflowException (p. 771) - If there is insufficient space in this buffer

ReadOnlyBufferException (p. 2520) - If this buffer is read-only

6.420.3.20 `LongBuffer& decaf::nio::LongBuffer::put (LongBuffer & src
) throw (BufferOverflowException, ReadOnlyBufferException,
lang::exceptions::IllegalArgumentException)`

This method transfers the long longs remaining in the given source buffer long longo this buffer.

If there are more long longs remaining in the source buffer than in this buffer, that is, if `src.remaining() > remaining()` (p. 746), then no long longs are transferred and a **BufferOverflowException** (p. 771) is thrown.

Otherwise, this method copies `n = src.remaining()` long longs from the given buffer long longo this buffer, starting at each buffer's current position. The positions of both buffers are then incremented by `n`.

Parameters

src - the buffer to take long longs from an place in this one.

Returns

a reference to this buffer

Exceptions

BufferOverflowException (p. 771) - If there is insufficient space in this buffer for the remaining long longs in the source buffer

IllegalArgumentException - If the source buffer is this buffer

ReadOnlyBufferException (p. 2520) - If this buffer is read-only

6.420.3.21 `virtual LongBuffer* decaf::nio::LongBuffer::slice () const [pure
virtual]`

Creates a new **LongBuffer** (p. 1955) whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the newly create **LongBuffer** (p. 1955) which the caller owns.

Implemented in `decaf::internal::nio::LongArrayBuffer` (p. 1955).

6.420.3.22 virtual std::string decaf::nio::LongBuffer::toString () const [virtual]**Returns**

a std::string describing this object

6.420.3.23 static LongBuffer* decaf::nio::LongBuffer::wrap (long long
* *array*, std::size_t *offset*, std::size_t *length*) throw (
lang::exceptions::NullPointerException) [static]

Wraps the passed buffer with a new **LongBuffer** (p. 1955).

The new buffer will be backed by the given long long array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be array.length, its position will be offset, its limit will be offset + length, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

array - The array that will back the new buffer

offset - The offset of the subarray to be used

length - The length of the subarray to be used

Returns

a new **LongBuffer** (p. 1955) that is backed by buffer, caller owns.

6.420.3.24 static LongBuffer* decaf::nio::LongBuffer::wrap (std::vector< long long
> & *buffer*) [static]

Wraps the passed STL long long Vector in a **LongBuffer** (p. 1955).

The new buffer will be backed by the given long long array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be buffer.size(), its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

buffer - The vector that will back the new buffer, the vector must have been sized to the desired size already by calling vector.resize(N).

Returns

a new **LongBuffer** (p. 1955) that is backed by buffer, caller owns.

The documentation for this class was generated from the following file:

- src/main/decaf/nio/**LongBuffer.h**

6.421 activemq::util::LongSequenceGenerator Class Reference

This class is used to generate a sequence of long long values that are incremented each time a new value is requested.

```
#include <src/main/activemq/util/LongSequenceGenerator.h>
```

Public Member Functions

- **LongSequenceGenerator** ()
- virtual **~LongSequenceGenerator** ()
- long long **getNextSequenceId** ()
- long long **getLastSequenceId** ()

6.421.1 Detailed Description

This class is used to generate a sequence of long long values that are incremented each time a new value is requested. This class is thread safe so the ids can be requested in different threads safely.

6.421.2 Constructor & Destructor Documentation

6.421.2.1 **activemq::util::LongSequenceGenerator::LongSequenceGenerator** ()

6.421.2.2 **virtual**
activemq::util::LongSequenceGenerator::~~LongSequenceGenerator ()
[inline, virtual]

6.421.3 Member Function Documentation

6.421.3.1 **long long activemq::util::LongSequenceGenerator::getLastSequenceId** ()

Returns

the last id that was generated.

6.421.3.2 **long long activemq::util::LongSequenceGenerator::getNextSequenceId** ()

Returns

the next id in the sequence.

The documentation for this class was generated from the following file:

- `src/main/activemq/util/LongSequenceGenerator.h`

6.422 decaf::net::MalformedURLException Class Reference

```
#include <src/main/decaf/net/MalformedURLException.h>
```

Inheritance diagram for decaf::net::MalformedURLException:

Public Member Functions

- **MalformedURLException** () throw ()
Default Constructor.
- **MalformedURLException** (const Exception &ex) throw ()
Conversion Constructor from some other Exception.
- **MalformedURLException** (const **MalformedURLException** &ex) throw ()
Copy Constructor.
- **MalformedURLException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **MalformedURLException** (const std::exception *cause) throw ()
Constructor.
- **MalformedURLException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **MalformedURLException** * clone () const
Clones this exception.
- virtual ~**MalformedURLException** () throw ()

6.422.1 Constructor & Destructor Documentation

6.422.1.1 decaf::net::MalformedURLException::MalformedURLException () throw () [inline]

Default Constructor.

6.422.1.2 decaf::net::MalformedURLException::MalformedURLException (const Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

Parameters

ex An exception that should become this type of Exception

References decaf::lang::Exception::Exception().

6.422.1.3 `decaf::net::MalformedURLException::MalformedURLException (const MalformedURLException & ex) throw () [inline]`

Copy Constructor.

Parameters

ex An exception that should become this type of Exception

References `decaf::lang::Exception::Exception()`.

6.422.1.4 `decaf::net::MalformedURLException::MalformedURLException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.422.1.5 `decaf::net::MalformedURLException::MalformedURLException (const std::exception * cause) throw () [inline]`

Constructor.

Parameters

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.422.1.6 `decaf::net::MalformedURLException::MalformedURLException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.422.1.7 virtual decaf::net::MalformedURLException::~~MalformedURLException
() throw () [inline, virtual]

6.422.2 Member Function Documentation

6.422.2.1 virtual MalformedURLException* decaf::net::MalformedURLException::clone () const
[inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::io::IOException** (p. 1709).

The documentation for this class was generated from the following file:

- src/main/decaf/net/MalformedURLException.h

6.423 decaf::util::Map< K, V, COMPARATOR > Class Template Reference

Map (p. 1970) template that wraps around a std::map to provide a more user-friendly interface and to provide common functions that do not exist in std::map.

```
#include <src/main/decaf/util/Map.h>
```

Inheritance diagram for decaf::util::Map< K, V, COMPARATOR >:

Data Structures

- class **Entry**

Public Member Functions

- **Map** ()
Default constructor - does nothing.
- virtual ~**Map** ()
- virtual bool **equals** (const **Map** &source) const =0
Comparison, equality is dependent on the method of determining if the element are equal.
- virtual void **copy** (const **Map** &source)=0
Copies the content of the source map into this map.

- virtual void **clear** ()=0 throw (decaf::lang::exceptions::UnsupportedOperationException)
Removes all keys and values from this map.
- virtual bool **containsKey** (const K &key) const =0
Indicates whether or this map contains a value for the given key.
- virtual bool **containsValue** (const V &value) const =0
Indicates whether or this map contains a value for the given value, i.e.
- virtual bool **isEmpty** () const =0
- virtual std::size_t **size** () const =0
- virtual V & **get** (const K &key)=0 throw (lang::exceptions::NoSuchElementException)
*Gets the value mapped to the specified key in the **Map** (p. 1970).*
- virtual const V & **get** (const K &key) const =0 throw (lang::exceptions::NoSuchElementException)
*Gets the value mapped to the specified key in the **Map** (p. 1970).*
- virtual void **put** (const K &key, const V &value)=0 throw (decaf::lang::exceptions::UnsupportedOperationException)
Sets the value for the specified key.
- virtual void **putAll** (const Map< K, V, COMPARATOR > &other)=0 throw (decaf::lang::exceptions::UnsupportedOperationException)
*Stores a copy of the Mappings contained in the other **Map** (p. 1970) in this one.*
- virtual V **remove** (const K &key)=0 throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException)
Removes the value (key/value pair) for the specified key from the map, returns a copy of the value that was mapped to the key.
- virtual std::vector< K > **keySet** () const =0
*Returns a **Set** (p. 2729) view of the mappings contained in this map.*
- virtual std::vector< V > **values** () const =0

6.423.1 Detailed Description

```
template<typename K, typename V, typename COMPARATOR = std::less<K>>
class decaf::util::Map< K, V, COMPARATOR >
```

Map (p. 1970) template that wraps around a std::map to provide a more user-friendly interface and to provide common functions that do not exist in std::map.

6.423.2 Constructor & Destructor Documentation

6.423.2.1 `template<typename K, typename V, typename COMPARATOR =
std::less<K>> decaf::util::Map< K, V, COMPARATOR >::Map ()
[inline]`

Default constructor - does nothing.

6.423.2.2 `template<typename K, typename V, typename COMPARATOR =
std::less<K>> virtual decaf::util::Map< K, V, COMPARATOR >::~~Map
() [inline, virtual]`

6.423.3 Member Function Documentation

6.423.3.1 `template<typename K, typename V, typename
COMPARATOR = std::less<K>> virtual void decaf::util::Map<
K, V, COMPARATOR >::clear () throw (
decaf::lang::exceptions::UnsupportedOperationException) [pure
virtual]`

Removes all keys and values from this map.

Exceptions

UnsupportedOperationException if this map is unmodifiable.

Implemented in `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >` (p. 1029), `decaf::util::StlMap< K, V, COMPARATOR >` (p. 2850), `decaf::util::concurrent::ConcurrentStlMap< Pointer< TransactionId >, Pointer< TransactionState >, TransactionId::COMPARATOR >` (p. 1029), `decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >` (p. 1029), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >` (p. 1029), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >` (p. 1029), `decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >` (p. 1029), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >` (p. 1029), `decaf::util::StlMap< cms::Session *, SessionResolver * >` (p. 2850), `decaf::util::StlMap< std::string, WireFormatFactory * >` (p. 2850), `decaf::util::StlMap< std::string, PrimitiveValueNode >` (p. 2850), `decaf::util::StlMap< std::string, cms::Queue * >` (p. 2850), `decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR >` (p. 2850), `decaf::util::StlMap< std::string, CachedConsumer * >` (p. 2850), `decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >` (p. 2850), `decaf::util::StlMap< std::string, TransportFactory * >` (p. 2850), `decaf::util::StlMap< int, Pointer< Command > >` (p. 2850), `decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR >` (p. 2850), `decaf::util::StlMap< std::string, CachedProducer * >` (p. 2850), and `decaf::util::StlMap< std::string, cms::Topic * >` (p. 2850).

6.423.3.2 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual bool decaf::util::Map< K, V, COMPARATOR >::containsKey (const K & key) const [pure virtual]`

Indicates whether or this map contains a value for the given key.

Parameters

key The key to look up.

Returns

true if this map contains the value, otherwise false.

Implemented in `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >` (p.1030), `decaf::util::StlMap< K, V, COMPARATOR >` (p.2850), `decaf::util::concurrent::ConcurrentStlMap< Pointer< TransactionId >, Pointer< TransactionState >, TransactionId::COMPARATOR >` (p.1030), `decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >` (p.1030), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >` (p.1030), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >` (p.1030), `decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >` (p.1030), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >` (p.1030), `decaf::util::StlMap< cms::Session *, SessionResolver * >` (p.2850), `decaf::util::StlMap< std::string, WireFormatFactory * >` (p.2850), `decaf::util::StlMap< std::string, PrimitiveValueNode >` (p.2850), `decaf::util::StlMap< std::string, cms::Queue * >` (p.2850), `decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR >` (p.2850), `decaf::util::StlMap< std::string, CachedConsumer * >` (p.2850), `decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >` (p.2850), `decaf::util::StlMap< std::string, TransportFactory * >` (p.2850), `decaf::util::StlMap< int, Pointer< Command > >` (p.2850), `decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR >` (p.2850), `decaf::util::StlMap< std::string, CachedProducer * >` (p.2850), and `decaf::util::StlMap< std::string, cms::Topic * >` (p.2850).

6.423.3.3 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual bool decaf::util::Map< K, V, COMPARATOR >::containsValue (const V & value) const [pure virtual]`

Indicates whether or this map contains a value for the given value, i.e.

they are equal, this is done by operator== so the types must pass equivalence testing in this manner.

Parameters

value The Value to look up.

Returns

true if this map contains the value, otherwise false.

Implemented in `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >` (p. 1030), `decaf::util::StlMap< K, V, COMPARATOR >` (p. 2850), `decaf::util::concurrent::ConcurrentStlMap< Pointer< TransactionId >, Pointer< TransactionState >, TransactionId::COMPARATOR >` (p. 1030), `decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >` (p. 1030), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >` (p. 1030), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >` (p. 1030), `decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >` (p. 1030), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >` (p. 1030), `decaf::util::StlMap< cms::Session *, SessionResolver * >` (p. 2850), `decaf::util::StlMap< std::string, WireFormatFactory * >` (p. 2850), `decaf::util::StlMap< std::string, PrimitiveValueNode >` (p. 2850), `decaf::util::StlMap< std::string, cms::Queue * >` (p. 2850), `decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR >` (p. 2850), `decaf::util::StlMap< std::string, CachedConsumer * >` (p. 2850), `decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >` (p. 2850), `decaf::util::StlMap< std::string, TransportFactory * >` (p. 2850), `decaf::util::StlMap< int, Pointer< Command > >` (p. 2850), `decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR >` (p. 2850), `decaf::util::StlMap< std::string, CachedProducer * >` (p. 2850), and `decaf::util::StlMap< std::string, cms::Topic * >` (p. 2850).

6.423.3.4 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::Map< K, V, COMPARATOR >::copy (const Map< K, V, COMPARATOR > & source)` [pure virtual]

Copies the content of the source map into this map.

Erases all existing data in this map.

Parameters

source The source object to copy from.

Implemented in `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >` (p. 1031), and `decaf::util::StlMap< K, V, COMPARATOR >` (p. 2851).

6.423.3.5 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual bool decaf::util::Map< K, V, COMPARATOR >::equals (const Map< K, V, COMPARATOR > & source) const` [pure virtual]

Comparison, equality is dependent on the method of determining if the element are equal.

Parameters

source - `Map` (p. 1970) to compare to this one.

Returns

true if the **Map** (p. 1970) passed is equal in value to this one.

Implemented in **decaf::util::concurrent::ConcurrentStlMap**< **K**, **V**, **COMPARATOR** > (p. 1031), and **decaf::util::StlMap**< **K**, **V**, **COMPARATOR** > (p. 2851).

6.423.3.6 `template<typename K, typename V, typename COMPARATOR =
std::less<K>> virtual V& decaf::util::Map< K, V, COMPARATOR >::get
(const K & key) throw (lang::exceptions::NoSuchElementException)
[pure virtual]`

Gets the value mapped to the specified key in the **Map** (p. 1970).

If there is no element in the map whose key is equivalent to the key provided then a **NoSuchElementException** is thrown.

Parameters

key The search key.

Returns

A reference to the value for the given key.

Exceptions

NoSuchElementException if the key requests doesn't exist in the **Map** (p. 1970).

Implemented in **decaf::util::concurrent::ConcurrentStlMap**< **K**, **V**, **COMPARATOR** > (p. 1031), **decaf::util::StlMap**< **K**, **V**, **COMPARATOR** > (p. 2851), **decaf::util::concurrent::ConcurrentStlMap**< **Pointer**< **TransactionId** >, **Pointer**< **TransactionState** >, **TransactionId::COMPARATOR** > (p. 1031), **decaf::util::concurrent::ConcurrentStlMap**< **Pointer**< **MessageId** >, **Pointer**< **Message** >, **MessageId::COMPARATOR** > (p. 1031), **decaf::util::concurrent::ConcurrentStlMap**< **Pointer**< **ConnectionId** >, **Pointer**< **ConnectionState** >, **ConnectionId::COMPARATOR** > (p. 1031), **decaf::util::concurrent::ConcurrentStlMap**< **Pointer**< **ConsumerId** >, **Pointer**< **ConsumerState** >, **ConsumerId::COMPARATOR** > (p. 1031), **decaf::util::concurrent::ConcurrentStlMap**< **Pointer**< **SessionId** >, **Pointer**< **SessionState** >, **SessionId::COMPARATOR** > (p. 1031), **decaf::util::concurrent::ConcurrentStlMap**< **Pointer**< **ProducerId** >, **Pointer**< **ProducerState** >, **ProducerId::COMPARATOR** > (p. 1031), **decaf::util::StlMap**< **cms::Session** *, **SessionResolver** * > (p. 2851), **decaf::util::StlMap**< **std::string**, **WireFormatFactory** * > (p. 2851), **decaf::util::StlMap**< **std::string**, **PrimitiveValueNode** > (p. 2851), **decaf::util::StlMap**< **std::string**, **cms::Queue** * > (p. 2851), **decaf::util::StlMap**< **Pointer**< **commands::ProducerId** >, **ActiveMQProducer** *, **commands::ProducerId::COMPARATOR** > (p. 2851), **decaf::util::StlMap**< **std::string**, **CachedConsumer** * > (p. 2851), **decaf::util::StlMap**< **Pointer**< **commands::ConsumerId** >, **ActiveMQConsumer** *, **commands::ConsumerId::COMPARATOR** > (p. 2851), **decaf::util::StlMap**< **std::string**, **TransportFactory** * > (p. 2851), **decaf::util::StlMap**< **int**, **Pointer**< **Command** > > (p. 2851), **decaf::util::StlMap**< **Pointer**< **commands::ConsumerId** >, **Dispatcher** *, **commands::ConsumerId::COMPARATOR** > (p. 2851), **decaf::util::StlMap**< **std::string**, **CachedProducer** * > (p. 2851), and **decaf::util::StlMap**< **std::string**, **cms::Topic** * > (p. 2851).

Referenced by `decaf::util::StlMap< std::string, cms::Topic * >::equals()`, and `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::equals()`.

6.423.3.7 `template<typename K, typename V, typename COMPARATOR
= std::less<K>> virtual const V& decaf::util::Map< K, V,
COMPARATOR >::get (const K & key) const throw (
lang::exceptions::NoSuchElementException) [pure virtual]`

Gets the value mapped to the specified key in the **Map** (p. 1970).

If there is no element in the map whose key is equivalent to the key provided then a `NoSuchElementException` is thrown.

Parameters

key The search key.

Returns

A {const} reference to the value for the given key.

Exceptions

NoSuchElementException if the key requests doesn't exist in the **Map** (p. 1970).

Implemented in `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >` (p. 1032), `decaf::util::StlMap< K, V, COMPARATOR >` (p. 2852), `decaf::util::concurrent::ConcurrentStlMap< Pointer< TransactionId >, Pointer< TransactionState >, TransactionId::COMPARATOR >` (p. 1032), `decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >` (p. 1032), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >` (p. 1032), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >` (p. 1032), `decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >` (p. 1032), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >` (p. 1032), `decaf::util::StlMap< cms::Session *, SessionResolver * >` (p. 2852), `decaf::util::StlMap< std::string, WireFormatFactory * >` (p. 2852), `decaf::util::StlMap< std::string, PrimitiveValueNode >` (p. 2852), `decaf::util::StlMap< std::string, cms::Queue * >` (p. 2852), `decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR >` (p. 2852), `decaf::util::StlMap< std::string, CachedConsumer * >` (p. 2852), `decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >` (p. 2852), `decaf::util::StlMap< std::string, TransportFactory * >` (p. 2852), `decaf::util::StlMap< int, Pointer< Command > >` (p. 2852), `decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR >` (p. 2852), `decaf::util::StlMap< std::string, CachedProducer * >` (p. 2852), and `decaf::util::StlMap< std::string, cms::Topic * >` (p. 2852).

6.423.3.8 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual bool decaf::util::Map< K, V, COMPARATOR >::isEmpty () const [pure virtual]`

Returns

if the **Map** (p. 1970) contains any element or not, TRUE or FALSE

Implemented in `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >` (p. 1032), `decaf::util::StlMap< K, V, COMPARATOR >` (p. 2852), `decaf::util::concurrent::ConcurrentStlMap< Pointer< TransactionId >, Pointer< TransactionState >, TransactionId::COMPARATOR >` (p. 1032), `decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >` (p. 1032), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >` (p. 1032), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >` (p. 1032), `decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >` (p. 1032), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >` (p. 1032), `decaf::util::StlMap< cms::Session *, SessionResolver * >` (p. 2852), `decaf::util::StlMap< std::string, WireFormatFactory * >` (p. 2852), `decaf::util::StlMap< std::string, PrimitiveValueNode >` (p. 2852), `decaf::util::StlMap< std::string, cms::Queue * >` (p. 2852), `decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR >` (p. 2852), `decaf::util::StlMap< std::string, CachedConsumer * >` (p. 2852), `decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >` (p. 2852), `decaf::util::StlMap< std::string, TransportFactory * >` (p. 2852), `decaf::util::StlMap< int, Pointer< Command > >` (p. 2852), `decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR >` (p. 2852), `decaf::util::StlMap< std::string, CachedProducer * >` (p. 2852), and `decaf::util::StlMap< std::string, cms::Topic * >` (p. 2852).

6.423.3.9 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual std::vector<K> decaf::util::Map< K, V, COMPARATOR >::keySet () const [pure virtual]`

Returns a **Set** (p. 2729) view of the mappings contained in this map.

The set is backed by the map, so changes to the map are reflected in the set, and vice-versa. If the map is modified while an iteration over the set is in progress (except through the iterator's own remove operation, or through the setValue operation on a map entry returned by the iterator) the results of the iteration are undefined. The set supports element removal, which removes the corresponding mapping from the map, via the **Iterator.remove** (p. 1718), **Set.remove** (p. 127), **removeAll**, **retainAll** and **clear** operations. It does not support the **add** or **addAll** operations.

Returns

the entire set of keys in this map as a `std::vector`.

Implemented in `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >` (p. 1032), `decaf::util::StlMap< K, V, COMPARATOR >`

(p. 2852), decaf::util::concurrent::ConcurrentStlMap< Pointer< TransactionId >, Pointer< TransactionState >, TransactionId::COMPARATOR > (p. 1032), decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR > (p. 1032), decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR > (p. 1032), decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR > (p. 1032), decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR > (p. 1032), decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR > (p. 1032), decaf::util::StlMap< cms::Session *, SessionResolver * > (p. 2852), decaf::util::StlMap< std::string, WireFormatFactory * > (p. 2852), decaf::util::StlMap< std::string, PrimitiveValueNode > (p. 2852), decaf::util::StlMap< std::string, cms::Queue * > (p. 2852), decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR > (p. 2852), decaf::util::StlMap< std::string, CachedConsumer * > (p. 2852), decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR > (p. 2852), decaf::util::StlMap< std::string, TransportFactory * > (p. 2852), decaf::util::StlMap< int, Pointer< Command > > (p. 2852), decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR > (p. 2852), decaf::util::StlMap< std::string, CachedProducer * > (p. 2852), and decaf::util::StlMap< std::string, cms::Topic * > (p. 2852).

Referenced by decaf::util::StlMap< std::string, cms::Topic * >::equals(), and decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::equals().

```
6.423.3.10 template<typename K, typename V, typename COMPARATOR =
std::less<K>> virtual void decaf::util::Map< K, V, COMPARATOR
>::put ( const K & key, const V & value ) throw (
decaf::lang::exceptions::UnsupportedOperationException ) [pure
virtual]
```

Sets the value for the specified key.

Parameters

key The target key.

value The value to be set.

Exceptions

UnsupportedOperationException if this map is unmodifiable.

Implemented in decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR > (p. 1034), decaf::util::StlMap< K, V, COMPARATOR > (p. 2854), decaf::util::concurrent::ConcurrentStlMap< Pointer< TransactionId >, Pointer< TransactionState >, TransactionId::COMPARATOR > (p. 1034), decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR > (p. 1034), decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR

> (p. 1034), decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR > (p. 1034), decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR > (p. 1034), decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR > (p. 1034), decaf::util::StlMap< cms::Session *, SessionResolver * > (p. 2854), decaf::util::StlMap< std::string, WireFormatFactory * > (p. 2854), decaf::util::StlMap< std::string, PrimitiveValueNode > (p. 2854), decaf::util::StlMap< std::string, cms::Queue * > (p. 2854), decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR > (p. 2854), decaf::util::StlMap< std::string, CachedConsumer * > (p. 2854), decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR > (p. 2854), decaf::util::StlMap< std::string, TransportFactory * > (p. 2854), decaf::util::StlMap< int, Pointer< Command > > (p. 2854), decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR > (p. 2854), decaf::util::StlMap< std::string, CachedProducer * > (p. 2854), and decaf::util::StlMap< std::string, cms::Topic * > (p. 2854).

6.423.3.11 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::Map< K, V, COMPARATOR >::putAll (const Map< K, V, COMPARATOR > & *other*) throw (decaf::lang::exceptions::UnsupportedOperationException) [pure virtual]

Stores a copy of the Mappings contained in the other **Map** (p. 1970) in this one.

Parameters

other A **Map** (p. 1970) instance whose elements are to all be inserted in this **Map** (p. 1970).

Exceptions

UnsupportedOperationException If the implementing class does not support the putAll operation.

Implemented in decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR > (p. 1034), decaf::util::StlMap< K, V, COMPARATOR > (p. 2854), decaf::util::concurrent::ConcurrentStlMap< Pointer< TransactionId >, Pointer< TransactionState >, TransactionId::COMPARATOR > (p. 1034), decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR > (p. 1034), decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR > (p. 1034), decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR > (p. 1034), decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR > (p. 1034), decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR > (p. 1034), decaf::util::StlMap< cms::Session *, SessionResolver * > (p. 2854), decaf::util::StlMap< std::string, WireFormatFactory * > (p. 2854), decaf::util::StlMap< std::string, PrimitiveValueNode > (p. 2854), decaf::util::StlMap< std::string, cms::Queue * > (p. 2854),

decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR > (p. 2854), decaf::util::StlMap< std::string, CachedConsumer * > (p. 2854), decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR > (p. 2854), decaf::util::StlMap< std::string, TransportFactory * > (p. 2854), decaf::util::StlMap< int, Pointer< Command > > (p. 2854), decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR > (p. 2854), decaf::util::StlMap< std::string, CachedProducer * > (p. 2854), and decaf::util::StlMap< std::string, cms::Topic * > (p. 2854).

6.423.3.12 template<typename K, typename V, typename COMPARATOR
= std::less<K>> virtual V decaf::util::Map< K, V,
COMPARATOR >::remove (const K & *key*) throw
(decaf::lang::exceptions::NoSuchElementException,
decaf::lang::exceptions::UnsupportedOperationException) [pure
virtual]

Removes the value (key/value pair) for the specified key from the map, returns a copy of the value that was mapped to the key.

Parameters

key The search key.

Returns

a copy of the element that was previously mapped to the given key

Exceptions

NoSuchElementException if this key is not in the Map (p. 1970).

UnsupportedOperationException if this map is unmodifiable.

Implemented in decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR > (p. 1036), decaf::util::StlMap< K, V, COMPARATOR > (p. 2855), decaf::util::concurrent::ConcurrentStlMap< Pointer< TransactionId >, Pointer< TransactionState >, TransactionId::COMPARATOR > (p. 1036), decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR > (p. 1036), decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR > (p. 1036), decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR > (p. 1036), decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR > (p. 1036), decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR > (p. 1036), decaf::util::StlMap< cms::Session *, SessionResolver * > (p. 2855), decaf::util::StlMap< std::string, WireFormatFactory * > (p. 2855), decaf::util::StlMap< std::string, PrimitiveValueNode > (p. 2855), decaf::util::StlMap< std::string, cms::Queue * > (p. 2855), decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR > (p. 2855), decaf::util::StlMap< std::string, CachedConsumer * > (p. 2855), decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR > (p. 2855),

`decaf::util::StlMap< std::string, TransportFactory * >` (p. 2855), `decaf::util::StlMap< int, Pointer< Command > >` (p. 2855), `decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR >` (p. 2855), `decaf::util::StlMap< std::string, CachedProducer * >` (p. 2855), and `decaf::util::StlMap< std::string, cms::Topic * >` (p. 2855).

6.423.3.13 `template<typename K, typename V, typename COMPARATOR`
`= std::less<K>> virtual std::size_t decaf::util::Map< K, V,`
`COMPARATOR >::size () const [pure virtual]`

Returns

The number of elements (key/value pairs) in this map.

Implemented in `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >` (p. 1037), `decaf::util::StlMap< K, V, COMPARATOR >` (p. 2855), `decaf::util::concurrent::ConcurrentStlMap< Pointer< TransactionId >, Pointer< TransactionState >, TransactionId::COMPARATOR >` (p. 1037), `decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >` (p. 1037), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >` (p. 1037), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >` (p. 1037), `decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >` (p. 1037), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >` (p. 1037), `decaf::util::StlMap< cms::Session *, SessionResolver * >` (p. 2855), `decaf::util::StlMap< std::string, WireFormatFactory * >` (p. 2855), `decaf::util::StlMap< std::string, PrimitiveValueNode >` (p. 2855), `decaf::util::StlMap< std::string, cms::Queue * >` (p. 2855), `decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR >` (p. 2855), `decaf::util::StlMap< std::string, CachedConsumer * >` (p. 2855), `decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >` (p. 2855), `decaf::util::StlMap< std::string, TransportFactory * >` (p. 2855), `decaf::util::StlMap< int, Pointer< Command > >` (p. 2855), `decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR >` (p. 2855), `decaf::util::StlMap< std::string, CachedProducer * >` (p. 2855), and `decaf::util::StlMap< std::string, cms::Topic * >` (p. 2855).

6.423.3.14 `template<typename K, typename V, typename COMPARATOR`
`= std::less<K>> virtual std::vector<V> decaf::util::Map< K, V,`
`COMPARATOR >::values () const [pure virtual]`

Returns

the entire set of values in this map as a `std::vector`.

Implemented in `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >` (p. 1038), `decaf::util::StlMap< K, V, COMPARATOR >` (p. 2856), `decaf::util::concurrent::ConcurrentStlMap< Pointer< TransactionId >, Pointer< TransactionState >, TransactionId::COMPARATOR`


```

> (p.1038), decaf::util::concurrent::ConcurrentStlMap< Pointer< Mes-
sageId >, Pointer< Message >, MessageId::COMPARATOR >
(p.1038), decaf::util::concurrent::ConcurrentStlMap< Pointer< Connec-
tionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR
> (p.1038), decaf::util::concurrent::ConcurrentStlMap< Pointer< Con-
sumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR
> (p.1038), decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId
>, Pointer< SessionState >, SessionId::COMPARATOR > (p.1038),
decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer<
ProducerState >, ProducerId::COMPARATOR > (p.1038), decaf::util::StlMap<
cms::Session *, SessionResolver * > (p.2856), decaf::util::StlMap< std::string,
WireFormatFactory * > (p.2856), decaf::util::StlMap< std::string, PrimitiveVal-
ueNode > (p.2856), decaf::util::StlMap< std::string, cms::Queue * > (p.2856),
decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, com-
mands::ProducerId::COMPARATOR > (p.2856), decaf::util::StlMap< std::string,
CachedConsumer * > (p.2856), decaf::util::StlMap< Pointer< commands::ConsumerId
>, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR > (p.2856),
decaf::util::StlMap< std::string, TransportFactory * > (p.2856), decaf::util::StlMap<
int, Pointer< Command > > (p.2856), decaf::util::StlMap< Pointer< com-
mands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR
> (p.2856), decaf::util::StlMap< std::string, CachedProducer * > (p.2856), and
decaf::util::StlMap< std::string, cms::Topic * > (p.2856).

```

The documentation for this class was generated from the following file:

- src/main/decaf/util/Map.h

6.424 cms::MapMessage Class Reference

A **MapMessage** (p.1982) object is used to send a set of name-value pairs.

```
#include <src/main/cms/MapMessage.h>
```

Inheritance diagram for cms::MapMessage:

Public Member Functions

- virtual **~MapMessage** ()
- virtual std::vector< std::string > **getMapNames** () const =0 throw (CMSEException)
*Returns an Enumeration of all the names in the **MapMessage** (p.1982) object.*
- virtual bool **itemExists** (const std::string &name) const =0 throw (CMSEException)
*Indicates whether an item exists in this **MapMessage** (p.1982) object.*
- virtual bool **getBoolean** (const std::string &name) const =0 throw (cms::MessageFormatException, cms::CMSEException)
Returns the Boolean value of the Specified name.
- virtual void **setBoolean** (const std::string &name, bool value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)

Sets a boolean value with the specified name into the Map.

- virtual unsigned char **getByte** (const std::string &name) const =0 throw (cms::MessageFormatException, cms::CMSEException)

Returns the Byte value of the Specified name.

- virtual void **setByte** (const std::string &name, unsigned char value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)

Sets a Byte value with the specified name into the Map.

- virtual std::vector< unsigned char > **getBytes** (const std::string &name) const =0 throw (cms::MessageFormatException, cms::CMSEException)

Returns the Bytes value of the Specified name.

- virtual void **setBytes** (const std::string &name, const std::vector< unsigned char > &value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)

Sets a Bytes value with the specified name into the Map.

- virtual char **getChar** (const std::string &name) const =0 throw (cms::MessageFormatException, cms::CMSEException)

Returns the Char value of the Specified name.

- virtual void **setChar** (const std::string &name, char value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)

Sets a Char value with the specified name into the Map.

- virtual double **getDouble** (const std::string &name) const =0 throw (cms::MessageFormatException, cms::CMSEException)

Returns the Double value of the Specified name.

- virtual void **setDouble** (const std::string &name, double value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)

Sets a Double value with the specified name into the Map.

- virtual float **getFloat** (const std::string &name) const =0 throw (cms::MessageFormatException, cms::CMSEException)

Returns the Float value of the Specified name.

- virtual void **setFloat** (const std::string &name, float value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)

Sets a Float value with the specified name into the Map.

- virtual int **getInt** (const std::string &name) const =0 throw (cms::MessageFormatException, cms::CMSEException)

Returns the Int value of the Specified name.

- virtual void **setInt** (const std::string &name, int value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)

Sets a Int value with the specified name into the Map.

- virtual long long **getLong** (const std::string &name) const =0 throw (cms::MessageFormatException, cms::CMSEException)
Returns the Long value of the Specified name.
- virtual void **setLong** (const std::string &name, long long value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)
Sets a Long value with the specified name into the Map.
- virtual short **getShort** (const std::string &name) const =0 throw (cms::MessageFormatException, cms::CMSEException)
Returns the Short value of the Specified name.
- virtual void **setShort** (const std::string &name, short value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)
Sets a Short value with the specified name into the Map.
- virtual std::string **getString** (const std::string &name) const =0 throw (cms::MessageFormatException, cms::CMSEException)
Returns the String value of the Specified name.
- virtual void **setString** (const std::string &name, const std::string &value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)
Sets a String value with the specified name into the Map.

6.424.1 Detailed Description

A **MapMessage** (p.1982) object is used to send a set of name-value pairs. The names are String objects, and the values are primitive data types in the Java programming language. The names must have a value that is not null, and not an empty string. The entries can be accessed sequentially or randomly by name. The order of the entries is undefined. **MapMessage** (p.1982) inherits from the **Message** (p.2036) interface and adds a message body that contains a Map.

When a client receives a **MapMessage** (p.1982), it is in read-only mode. If a client attempts to write to the message at this point, a **MessageNotWriteableException** (p.2188) is thrown. To place the **MapMessage** (p.1982) back into a state where it can be read from and written to, call the **clearBody** method.

MapMessage (p.1982) objects support the following conversion table. The marked cases must be supported. The unmarked cases must throw a **CMSEException** (p.960). The String-to-primitive conversions may throw a **MessageFormatException** (p.2141) if the primitive's **valueOf()** method does not accept it as a valid String representation of the primitive.

A value written as the row type can be read as the column type.

	boolean	byte	short	char	int	long	float	double	String	byte[]
boolean	X									X
byte		X	X		X	X				X
short			X		X	X				X
char				X						X
int					X	X				X
long						X				X
float							X	X		X

double								X	X
String		X	X	X		X	X	X	X
byte[]									X

Since

1.0

6.424.2 Constructor & Destructor Documentation

6.424.2.1 virtual cms::MapMessage::~MapMessage () [inline, virtual]

6.424.3 Member Function Documentation

6.424.3.1 virtual bool cms::MapMessage::getBoolean (const std::string & *name*) const throw (cms::MessageFormatException, cms::CMSException) [pure virtual]

Returns the Boolean value of the Specified name.

Parameters

name Name of the value to fetch from the map

Exceptions

CMSException (p. 960) - if the operation fails due to an internal error.

MessageFormatException - if this type conversion is invalid.

6.424.3.2 virtual unsigned char cms::MapMessage::getByte (const std::string & *name*) const throw (cms::MessageFormatException, cms::CMSException) [pure virtual]

Returns the Byte value of the Specified name.

Parameters

name Name of the value to fetch from the map

Exceptions

CMSException (p. 960) - if the operation fails due to an internal error.

MessageFormatException (p. 2141) - if this type conversion is invalid.

6.424.3.3 virtual std::vector<unsigned char> cms::MapMessage::getBytes (const std::string & *name*) const throw (cms::MessageFormatException, cms::CMSException) [pure virtual]

Returns the Bytes value of the Specified name.

Parameters

name Name of the value to fetch from the map

Exceptions

CMSEException (p. 960) - if the operation fails due to an internal error.

MessageFormatException (p. 2141) - if this type conversion is invalid.

6.424.3.4 virtual char cms::MapMessage::getChar (const std::string & *name*
) const throw (cms::MessageFormatException, cms::CMSEException)
[pure virtual]

Returns the Char value of the Specified name.

Parameters

name name of the value to fetch from the map

Exceptions

CMSEException (p. 960) - if the operation fails due to an internal error.

MessageFormatException (p. 2141) - if this type conversion is invalid.

6.424.3.5 virtual double cms::MapMessage::getDouble (const std::string & *name*
) const throw (cms::MessageFormatException, cms::CMSEException)
[pure virtual]

Returns the Double value of the Specified name.

Parameters

name Name of the value to fetch from the map

Exceptions

CMSEException (p. 960) - if the operation fails due to an internal error.

MessageFormatException (p. 2141) - if this type conversion is invalid.

6.424.3.6 virtual float cms::MapMessage::getFloat (const std::string & *name*
) const throw (cms::MessageFormatException, cms::CMSEException)
[pure virtual]

Returns the Float value of the Specified name.

Parameters

name Name of the value to fetch from the map

Exceptions

CMSEException (p. 960) - if the operation fails due to an internal error.

MessageFormatException (p. 2141) - if this type conversion is invalid.

6.424.3.7 `virtual int cms::MapMessage::getInt (const std::string & name) const
throw (cms::MessageFormatException, cms::CMSException) [pure
virtual]`

Returns the Int value of the Specified name.

Parameters

name Name of the value to fetch from the map

Exceptions

CMSException (p. 960) - if the operation fails due to an internal error.

MessageFormatException (p. 2141) - if this type conversion is invalid.

6.424.3.8 `virtual long long cms::MapMessage::getLong (const std::string & name
) const throw (cms::MessageFormatException, cms::CMSException)
[pure virtual]`

Returns the Long value of the Specified name.

Parameters

name Name of the value to fetch from the map

Exceptions

CMSException (p. 960) - if the operation fails due to an internal error.

MessageFormatException (p. 2141) - if this type conversion is invalid.

6.424.3.9 `virtual std::vector< std::string > cms::MapMessage::getMapNames ()
const throw (CMSException) [pure virtual]`

Returns an Enumeration of all the names in the **MapMessage** (p. 1982) object.

Returns

STL Vector of String values, each of which is the name of an item in the **MapMessage** (p. 1982)

Exceptions

CMSException (p. 960) - if the operation fails due to an internal error.

6.424.3.10 `virtual short cms::MapMessage::getShort (const std::string & name)
const throw (cms::MessageFormatException, cms::CMSException)
[pure virtual]`

Returns the Short value of the Specified name.

Parameters

name Name of the value to fetch from the map

Exceptions

CMSEException (p. 960) - if the operation fails due to an internal error.

MessageFormatException (p. 2141) - if this type conversion is invalid.

6.424.3.11 virtual std::string cms::MapMessage::getString (const std::string & *name*) const throw (cms::MessageFormatException, cms::CMSEException) [pure virtual]

Returns the String value of the Specified name.

Parameters

name Name of the value to fetch from the map

Exceptions

CMSEException (p. 960) - if the operation fails due to an internal error.

MessageFormatException (p. 2141) - if this type conversion is invalid.

6.424.3.12 virtual bool cms::MapMessage::itemExists (const std::string & *name*) const throw (CMSEException) [pure virtual]

Indicates whether an item exists in this **MapMessage** (p.1982) object.

Parameters

name String name of the Object in question

Returns

boolean value indicating if the name is in the map

Exceptions

CMSEException (p. 960) - if the operation fails due to an internal error.

6.424.3.13 virtual void cms::MapMessage::setBoolean (const std::string & *name*, bool *value*) throw (cms::MessageNotWriteableException, cms::CMSEException) [pure virtual]

Sets a boolean value with the specified name into the Map.

Parameters

name the name of the boolean

value the boolean value to set in the Map

Exceptions

CMSEException (p. 960) - if the operation fails due to an internal error.

MessageNotWritableException - if the **Message** (p.2036) is in Read-only Mode.

6.424.3.14 `virtual void cms::MapMessage::setByte (const std::string & name,
unsigned char value) throw (cms::MessageNotWriteableException,
cms::CMSException) [pure virtual]`

Sets a Byte value with the specified name into the Map.

Parameters

name the name of the Byte

value the Byte value to set in the Map

Exceptions

CMSException (p. 960) - if the operation fails due to an internal error.

MessageNotWriteableException (p. 2188) - if the **Message** (p. 2036) is in Read-only Mode.

6.424.3.15 `virtual void cms::MapMessage::setBytes (const std::string &
name, const std::vector< unsigned char > & value) throw (cms::MessageNotWriteableException, cms::CMSException) [pure
virtual]`

Sets a Bytes value with the specified name into the Map.

Parameters

name The name of the Bytes

value The Bytes value to set in the Map

Exceptions

CMSException (p. 960) - if the operation fails due to an internal error.

MessageNotWriteableException (p. 2188) - if the **Message** (p. 2036) is in Read-only Mode.

6.424.3.16 `virtual void cms::MapMessage::setChar (const std::string &
name, char value) throw (cms::MessageNotWriteableException,
cms::CMSException) [pure virtual]`

Sets a Char value with the specified name into the Map.

Parameters

name the name of the Char

value the Char value to set in the Map

Exceptions

CMSException (p. 960) - if the operation fails due to an internal error.

MessageNotWriteableException (p. 2188) - if the **Message** (p. 2036) is in Read-only Mode.

6.424.3.17 `virtual void cms::MapMessage::setDouble (const std::string & name, double value) throw (cms::MessageNotWriteableException, cms::CMSException) [pure virtual]`

Sets a Double value with the specified name into the Map.

Parameters

name The name of the Double

value The Double value to set in the Map

Exceptions

CMSException (p. 960) - if the operation fails due to an internal error.

MessageNotWriteableException (p. 2188) - if the **Message** (p. 2036) is in Read-only Mode.

6.424.3.18 `virtual void cms::MapMessage::setFloat (const std::string & name, float value) throw (cms::MessageNotWriteableException, cms::CMSException) [pure virtual]`

Sets a Float value with the specified name into the Map.

Parameters

name The name of the Float

value The Float value to set in the Map

Exceptions

CMSException (p. 960) - if the operation fails due to an internal error.

MessageNotWriteableException (p. 2188) - if the **Message** (p. 2036) is in Read-only Mode.

6.424.3.19 `virtual void cms::MapMessage::setInt (const std::string & name, int value) throw (cms::MessageNotWriteableException, cms::CMSException) [pure virtual]`

Sets a Int value with the specified name into the Map.

Parameters

name The name of the Int

value The Int value to set in the Map

Exceptions

CMSException (p. 960) - if the operation fails due to an internal error.

MessageNotWriteableException (p. 2188) - if the **Message** (p. 2036) is in Read-only Mode.

6.424.3.20 `virtual void cms::MapMessage::setLong (const std::string & name,
long long value) throw (cms::MessageNotWriteableException,
cms::CMSException) [pure virtual]`

Sets a Long value with the specified name into the Map.

Parameters

name The name of the Long
value The Long value to set in the Map

Exceptions

CMSException (p. 960) - if the operation fails due to an internal error.
MessageNotWriteableException (p. 2188) - if the **Message** (p. 2036) is in Read-only Mode.

6.424.3.21 `virtual void cms::MapMessage::setShort (const std::string &
name, short value) throw (cms::MessageNotWriteableException,
cms::CMSException) [pure virtual]`

Sets a Short value with the specified name into the Map.

Parameters

name The name of the Short
value The Short value to set in the Map

Exceptions

CMSException (p. 960) - if the operation fails due to an internal error.
MessageNotWriteableException (p. 2188) - if the **Message** (p. 2036) is in Read-only Mode.

6.424.3.22 `virtual void cms::MapMessage::setString (const std::string
& name, const std::string & value) throw (
cms::MessageNotWriteableException, cms::CMSException) [pure
virtual]`

Sets a String value with the specified name into the Map.

Parameters

name The name of the String
value The String value to set in the Map

Exceptions

CMSException (p. 960) - if the operation fails due to an internal error.
MessageNotWriteableException (p. 2188) - if the **Message** (p. 2036) is in Read-only Mode.

The documentation for this class was generated from the following file:

- `src/main/cms/MapMessage.h`

6.425 decaf::util::logging::MarkBlockLogger Class Reference

Defines a class that can be used to mark the entry and exit from scoped blocks.

```
#include <src/main/decaf/util/logging/MarkBlockLogger.h>
```

Public Member Functions

- **MarkBlockLogger** (**Logger** *logger, const std::string &blockName)

Constructor - Marks Block entry.

- virtual ~**MarkBlockLogger** ()

6.425.1 Detailed Description

Defines a class that can be used to mark the entry and exit from scoped blocks. Create an instance of this class at the start of a scoped block, passing it the logger to use and the name of the block. The block entry and exit will be marked using the scope name, logger to the logger at the MARKBLOCK log level.

6.425.2 Constructor & Destructor Documentation

- 6.425.2.1 decaf::util::logging::MarkBlockLogger::MarkBlockLogger (Logger *
logger, const std::string & blockName) [inline]**

Constructor - Marks Block entry.

Parameters

logger **Logger** (p.1908) to use

blockName Block name

- 6.425.2.2 virtual decaf::util::logging::MarkBlockLogger::~MarkBlockLogger ()
[inline, virtual]**

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/MarkBlockLogger.h

6.426 activemq::wireformat::MarshalAware Class Reference

```
#include <src/main/activemq/wireformat/MarshalAware.h>
```

Inheritance diagram for activemq::wireformat::MarshalAware:

Public Member Functions

- virtual `~MarshalAware ()`
- virtual bool `isMarshalAware () const =0`
Determine if the class implementing this interface is really wanting to be told about marshaling.
- virtual void `beforeMarshal (WireFormat *wireFormat)=0 throw (decaf::io::IOException)`
Called before marshaling is started to prepare the object to be marshaled.
- virtual void `afterMarshal (WireFormat *wireFormat)=0 throw (decaf::io::IOException)`
Called after marshaling is started to cleanup the object being marshaled.
- virtual void `beforeUnmarshal (WireFormat *wireFormat)=0 throw (decaf::io::IOException)`
Called before unmarshaling is started to prepare the object to be unmarshaled.
- virtual void `afterUnmarshal (WireFormat *wireFormat)=0 throw (decaf::io::IOException)`
Called after unmarshaling is started to cleanup the object being unmarshaled.
- virtual void `setMarshaledForm (WireFormat *wireFormat, const std::vector< char > &data)=0`
Called to set the data to this object that will contain the objects marshaled form.
- virtual `std::vector< unsigned char > getMarshaledForm (WireFormat *wireFormat)=0`
Called to get the data to this object that will contain the objects marshaled form.

6.426.1 Constructor & Destructor Documentation

- 6.426.1.1 `virtual activemq:wireformat::MarshalAware::~~MarshalAware ()`
`[inline, virtual]`

6.426.2 Member Function Documentation

- 6.426.2.1 `virtual void activemq:wireformat::MarshalAware::afterMarshal (WireFormat * wireFormat) throw (decaf::io::IOException)` `[pure virtual]`

Called after marshaling is started to cleanup the object being marshaled.

Parameters

wireFormat - the wireformat object to control marshaling

6.426.2.2 virtual void activemq::wireformat::MarshalAware::afterUnmarshal (WireFormat * *wireFormat*) throw (decaf::io::IOException) [pure virtual]

Called after unmarshaling is started to cleanup the object being unmarshaled.

Parameters

wireFormat - the wireformat object to control unmarshaling

6.426.2.3 virtual void activemq::wireformat::MarshalAware::beforeMarshal (WireFormat * *wireFormat*) throw (decaf::io::IOException) [pure virtual]

Called before marshaling is started to prepare the object to be marshaled.

Parameters

wireFormat - the wireformat object to control marshaling

Implemented in `activemq::commands::ActiveMQMapMessage` (p. 275), and `activemq::commands::ActiveMQTextMessage` (p. 526).

6.426.2.4 virtual void activemq::wireformat::MarshalAware::beforeUnmarshal (WireFormat * *wireFormat*) throw (decaf::io::IOException) [pure virtual]

Called before unmarshaling is started to prepare the object to be unmarshaled.

Parameters

wireFormat - the wireformat object to control unmarshaling

6.426.2.5 virtual std::vector<unsigned char> activemq::wireformat::MarshalAware::getMarshaledForm (WireFormat * *wireFormat*) [pure virtual]

Called to get the data to this object that will contain the objects marshaled form.

Parameters

wireFormat - the wireformat object to control unmarshaling

Returns

buffer that holds the objects data.

6.426.2.6 `virtual bool activemq::wireformat::MarshalAware::isMarshalAware ()` `const [pure virtual]`

Determine if the class implementing this interface is really wanting to be told about marshaling.

Normally if you didn't want to be marshal aware you just wouldn't implement this interface but since this is C++ and we don't have true interfaces we need a flat inheritance hierarchy, so we always implement this.

Returns

true if this class cares about marshaling.

Implemented in `activemq::commands::ActiveMQMapMessage` (p. 280), `activemq::commands::BaseDataStructure` (p. 662), `activemq::commands::Message` (p. 2029), and `activemq::commands::WireFormatInfo` (p. 3158).

6.426.2.7 `virtual void activemq::wireformat::MarshalAware::setMarshaledForm (` `WireFormat * wireFormat, const std::vector< char > & data) [pure` `virtual]`

Called to set the data to this object that will contain the objects marshaled form.

Parameters

wireFormat - the wireformat object to control unmarshaling

data - vector of object binary data

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/MarshalAware.h`

6.427 `activemq::wireformat::openwire::marshal::v2::MarshallerFactory` Class Reference

Used to create marshallers for a specific version of the wire protocol.

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/MarshallerFactory.h>
```

Public Member Functions

- `virtual ~MarshallerFactory ()`
- `virtual void configure (OpenWireFormat *format)`

6.427.1 Detailed Description

Used to create marshallers for a specific version of the wire protocol. NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Groovy scripts in the `activemq-openwire-generator` module

6.427.2 Constructor & Destructor Documentation

6.427.2.1 virtual
activemq::wireformat::openwire::marshal::v2::MarshallerFactory::~~MarshallerFactory
() [inline, virtual]

6.427.3 Member Function Documentation

6.427.3.1 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::MarshallerFactory::configure
(OpenWireFormat * *format*) [virtual]

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/MarshallerFactory.h

6.428 activemq::wireformat::openwire::marshal::v4::MarshallerFactory Class Reference

Used to create marshallers for a specific version of the wire protocol.

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/MarshallerFactory.h>
```

Public Member Functions

- virtual ~MarshallerFactory ()
- virtual void configure (OpenWireFormat *format)

6.428.1 Detailed Description

Used to create marshallers for a specific version of the wire protocol. NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Groovy scripts in the activemq-openwire-generator module

6.428.2 Constructor & Destructor Documentation

6.428.2.1 virtual
activemq::wireformat::openwire::marshal::v4::MarshallerFactory::~~MarshallerFactory
() [inline, virtual]

6.428.3 Member Function Documentation

6.428.3.1 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::MarshallerFactory::configure
(OpenWireFormat * *format*) [virtual]

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/MarshallerFactory.h

6.429 activemq::wireformat::openwire::marshal::v1::MarshallerFactory Class Reference

Used to create marshallers for a specific version of the wire protocol.

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/MarshallerFactory.h>
```

Public Member Functions

- virtual `~MarshallerFactory()`
- virtual void `configure(OpenWireFormat *format)`

6.429.1 Detailed Description

Used to create marshallers for a specific version of the wire protocol. NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Groovy scripts in the activemq-openwire-generator module

6.429.2 Constructor & Destructor Documentation

6.429.2.1 virtual
`activemq::wireformat::openwire::marshal::v1::MarshallerFactory::~~MarshallerFactory()` [inline, virtual]

6.429.3 Member Function Documentation

6.429.3.1 virtual void `activemq::wireformat::openwire::marshal::v1::MarshallerFactory::configure(OpenWireFormat * format)` [virtual]

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/MarshallerFactory.h`

6.430 activemq::wireformat::openwire::marshal::v3::MarshallerFactory Class Reference

Used to create marshallers for a specific version of the wire protocol.

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/MarshallerFactory.h>
```

Public Member Functions

- virtual `~MarshallerFactory()`
- virtual void `configure(OpenWireFormat *format)`

6.430.1 Detailed Description

Used to create marshallers for a specific version of the wire protocol. NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Groovy scripts in the activemq-openwire-generator module

6.430.2 Constructor & Destructor Documentation

6.430.2.1 virtual
activemq::wireformat::openwire::marshal::v3::MarshallerFactory::~~MarshallerFactory
() [inline, virtual]

6.430.3 Member Function Documentation

6.430.3.1 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::MarshallerFactory::configure
(OpenWireFormat * *format*) [virtual]

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**MarshallerFactory.h**

6.431 activemq::wireformat::openwire::marshal::v5::MarshallerFactory Class Reference

Used to create marshallers for a specific version of the wire protocol.

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/MarshallerFactory.h>
```

Public Member Functions

- virtual ~MarshallerFactory ()
- virtual void **configure** (OpenWireFormat *format)

6.431.1 Detailed Description

Used to create marshallers for a specific version of the wire protocol. NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Groovy scripts in the activemq-openwire-generator module

6.431.2 Constructor & Destructor Documentation

6.431.2.1 `virtual
activemq::wireformat::openwire::marshal::v5::MarshallerFactory::~MarshallerFactory
() [inline, virtual]`

6.431.3 Member Function Documentation

6.431.3.1 `virtual void ac-
tivemq::wireformat::openwire::marshal::v5::MarshallerFactory::configure
(OpenWireFormat * format) [virtual]`

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/MarshallerFactory.h`

6.432 decaf::lang::Math Class Reference

The class `Math` (p. 1999) contains methods for performing basic numeric operations such as the elementary exponential, logarithm, square root, and trigonometric functions.

`#include <src/main/decaf/lang/Math.h>`

Public Member Functions

- `Math ()`
- `virtual ~Math ()`

Static Public Member Functions

- static int **abs** (int value)
Returns the absolute value of an int value.
- static long long **abs** (long long value)
Returns the absolute value of an long long value.
- static float **abs** (float value)
Returns the absolute value of a float value.
- static double **abs** (double value)
Returns the absolute value of a double value.
- static double **sqrt** (double value)
Returns the arc cosine of an angle, in the range of 0.0 through pi.
- static double **pow** (double base, double exp)
Returns the value of the first argument raised to the power of the second argument.
- static short **min** (short a, short b)

Returns the double value that is closest in value to the argument and is equal to a mathematical integer.

- static int **min** (int a, int b)
*Returns the smaller of two **int** values.*
- static int **min** (unsigned int a, unsigned int b)
*Returns the smaller of two **unsigned int** values.*
- static long long **min** (long long a, long long b)
*Returns the smaller of two **long long** values.*
- static float **min** (float a, float b)
Returns the smaller of two float values.
- static double **min** (double a, double b)
Returns the smaller of two double values.
- static short **max** (short a, short b)
*Returns the larger of two **short** values.*
- static int **max** (int a, int b)
*Returns the larger of two **int** values.*
- static long long **max** (long long a, long long b)
*Returns the larger of two **long long** values.*
- static float **max** (float a, float b)
Returns the greater of two float values.
- static double **max** (double a, double b)
Returns the greater of two double values.
- static double **ceil** (double value)
Returns the natural logarithm (base e) of a double value.
- static double **floor** (double value)
Returns the largest (closest to positive infinity) double value that is less than or equal to the argument and is equal to a mathematical integer.
- static int **round** (float value)
Returns the closest int to the argument.
- static long long **round** (double value)
Returns the closest long long to the argument.
- static double **random** ()
Computes the remainder operation on two arguments as prescribed by the IEEE 754 standard.
- static float **signum** (float value)

Returns Euler's number e raised to the power of a double value.

- static double **signum** (double value)

Returns the signum function of the argument; zero if the argument is zero, 1.0f if the argument is greater than zero, -1.0f if the argument is less than zero.

- static double **toRadians** (double angdeg)

Returns the measure in radians of the supplied degree angle.

- static double **toDegrees** (double angrad)

Returns the measure in degrees of the supplied radian angle.

Static Public Attributes

- static const double **E**
- static const double **PI**

6.432.1 Detailed Description

The class `Math` (p. 1999) contains methods for performing basic numeric operations such as the elementary exponential, logarithm, square root, and trigonometric functions.

6.432.2 Constructor & Destructor Documentation

6.432.2.1 `decaf::lang::Math::Math ()` [inline]

6.432.2.2 `virtual decaf::lang::Math::~~Math ()` [inline, virtual]

6.432.3 Member Function Documentation

6.432.3.1 `static int decaf::lang::Math::abs (int value)` [inline, static]

Returns the absolute value of an int value.

If the argument is not negative, the argument is returned. If the argument is negative, the negation of the argument is returned.

Parameters

value - the value to return the abs of

Returns

the value if positive, otherwise the negative of value

6.432.3.2 `static long long decaf::lang::Math::abs (long long value)` [inline, static]

Returns the absolute value of an long long value.

If the argument is not negative, the argument is returned. If the argument is negative, the negation of the argument is returned.

Parameters

value - the value to return the abs of

Returns

the value if positive, otherwise the negative of value

6.432.3.3 static double decaf::lang::Math::abs (double *value*) [static]

Returns the absolute value of a double value.

If the argument is not negative, the argument is returned. If the argument is negative, the negation of the argument is returned. Special cases:

o If the argument is positive zero or negative zero, the result is positive zero. o If the argument is infinite, the result is positive infinity. o If the argument is NaN, the result is NaN.

In other words, the result is the same as the value of the expression: **Double::longBitsToDouble** (p. 1447)(0x7fffffffffffULL & Double::doubleToLongBits(*value*))

Parameters

value - the value to return the abs of

Returns

the value if positive, otherwise the negative of value

6.432.3.4 static float decaf::lang::Math::abs (float *value*) [static]

Returns the absolute value of a float value.

If the argument is not negative, the argument is returned. If the argument is negative, the negation of the argument is returned. Special cases:

o If the argument is positive zero or negative zero, the result is positive zero. o If the argument is infinite, the result is positive infinity. o If the argument is NaN, the result is NaN.

In other words, the result is the same as the value of the expression: **Float::intBitsToFloat** (p. 1549)(0x7fffffff & Float::floatToIntBits(*value*))

Parameters

value - the value to return the abs of

Returns

the value if positive, otherwise the negative of value

6.432.3.5 static double decaf::lang::Math::ceil (double *value*) [static]

Returns the natural logarithm (base e) of a double value.

Special cases:

o If the argument is NaN or less than zero, then the result is NaN. o If the argument is positive infinity, then the result is positive infinity. o If the argument is positive zero or negative zero, then the result is negative infinity.

Parameters

value the value to compute the natural log of.

Returns

the natural log of value. Returns the base 10 logarithm of a double value. Special cases:

o If the argument is NaN or less than zero, then the result is NaN. o If the argument is positive infinity, then the result is positive infinity. o If the argument is positive zero or negative zero, then the result is negative infinity. o If the argument is equal to 10^n for integer n , then the result is n .

Parameters

value - the value to operate on

Returns

the long base 10 of value Returns the natural logarithm of the sum of the argument and 1. Note that for small values x , the result of $\log_{10}(x)$ is much closer to the true result of $\ln(1 + x)$ than the floating-point evaluation of $\log(1.0+x)$.

Special cases:

o If the argument is NaN or less than -1, then the result is NaN. o If the argument is positive infinity, then the result is positive infinity. o If the argument is negative one, then the result is negative infinity. o If the argument is zero, then the result is a zero with the same sign as the argument.

Parameters

value - the value to operate on

Returns

the the value $\ln(x + 1)$, the natural log of $x + 1$ Returns the smallest (closest to negative infinity) double value that is greater than or equal to the argument and is equal to a mathematical integer. Special cases:

o If the argument value is already equal to a mathematical integer, then the result is the same as the argument. o If the argument is NaN or an infinity or positive zero or negative zero, then the result is the same as the argument. o If the argument value is less than zero but greater than -1.0, then the result is negative zero.

Note that the value of `Math.ceil(x)` is exactly the value of `-Math.floor(-x)`.

Parameters

value - the value to find the ceiling of

Returns

the smallest (closest to negative infinity) floating-point value that is greater than or equal to the argument and is equal to a mathematical integer.

6.432.3.6 static double decaf::lang::Math::floor (double *value*) [static]

Returns the largest (closest to positive infinity) double value that is less than or equal to the argument and is equal to a mathematical integer.

Special cases:

- o If the argument value is already equal to a mathematical integer, then the result is the same as the argument.
- o If the argument is NaN or an infinity or positive zero or negative zero, then the result is the same as the argument.

Parameters

value - the value to find the floor of

Returns

the largest (closest to positive infinity) floating-point value that less than or equal to the argument and is equal to a mathematical integer.

6.432.3.7 static float decaf::lang::Math::max (float *a*, float *b*) [static]

Returns the greater of two float values.

That is, the result is the argument closer to positive infinity. If the arguments have the same value, the result is that same value. If either value is NaN, then the result is NaN. Unlike the numerical comparison operators, this method considers negative zero to be strictly smaller than positive zero. If one argument is positive zero and the other negative zero, the result is positive zero.

Parameters

- a* - an argument.
- b* - another argument.

Returns

the larger of *a* and *b*.

6.432.3.8 static double decaf::lang::Math::max (double *a*, double *b*) [static]

Returns the greater of two double values.

That is, the result is the argument closer to positive infinity. If the arguments have the same value, the result is that same value. If either value is NaN, then the result is NaN. Unlike the numerical comparison operators, this method considers negative zero to be strictly smaller than positive zero. If one argument is positive zero and the other negative zero, the result is positive zero.

Parameters

- a* - an argument.
- b* - another argument.

Returns

the larger of *a* and *b*.

6.432.3.9 `static short decaf::lang::Math::max (short a, short b) [inline, static]`

Returns the larger of two `short` values.

That is, the result the argument closer to the value of `Short::MAX_VALUE` (p.2738). If the arguments have the same value, the result is that same value.

Parameters

a - an argument.

b - another argument.

Returns

the larger of *a* and *b*.

6.432.3.10 `static int decaf::lang::Math::max (int a, int b) [inline, static]`

Returns the larger of two `int` values.

That is, the result the argument closer to the value of `Integer::MAX_VALUE` (p.1666). If the arguments have the same value, the result is that same value.

Parameters

a - an argument.

b - another argument.

Returns

the larger of *a* and *b*.

6.432.3.11 `static long long decaf::lang::Math::max (long long a, long long b) [inline, static]`

Returns the larger of two `long long` values.

That is, the result the argument closer to the value of `Long::MAX_VALUE` (p.1947). If the arguments have the same value, the result is that same value.

Parameters

a - an argument.

b - another argument.

Returns

the larger of *a* and *b*.

6.432.3.12 static int decaf::lang::Math::min (int *a*, int *b*) [inline, static]

Returns the smaller of two `int` values.

That is, the result the argument closer to the value of `Integer::MIN_VALUE` (p.1666). If the arguments have the same value, the result is that same value.

Parameters

a - an argument.

b - another argument.

Returns

the smaller of `a` and `b`.

6.432.3.13 static int decaf::lang::Math::min (unsigned int *a*, unsigned int *b*) [inline, static]

Returns the smaller of two `unsigned int` values.

That is, the result the argument closer to the value of `Integer::MIN_VALUE` (p.1666). If the arguments have the same value, the result is that same value.

Parameters

a - an argument.

b - another argument.

Returns

the smaller of `a` and `b`.

6.432.3.14 static long long decaf::lang::Math::min (long long *a*, long long *b*) [inline, static]

Returns the smaller of two `long long` values.

That is, the result the argument closer to the value of `Long::MIN_VALUE` (p.1947). If the arguments have the same value, the result is that same value.

Parameters

a - an argument.

b - another argument.

Returns

the smaller of `a` and `b`.

6.432.3.15 `static float decaf::lang::Math::min (float a, float b) [static]`

Returns the smaller of two float values.

That is, the result is the value closer to negative infinity. If the arguments have the same value, the result is that same value. If either value is NaN, then the result is NaN. Unlike the numerical comparison operators, this method considers negative zero to be strictly smaller than positive zero. If one argument is positive zero and the other is negative zero, the result is negative zero.

Parameters

- a* - an argument.
- b* - another argument.

Returns

the smaller of *a* and *b*.

6.432.3.16 `static double decaf::lang::Math::min (double a, double b) [static]`

Returns the smaller of two double values.

That is, the result is the value closer to negative infinity. If the arguments have the same value, the result is that same value. If either value is NaN, then the result is NaN. Unlike the numerical comparison operators, this method considers negative zero to be strictly smaller than positive zero. If one argument is positive zero and the other is negative zero, the result is negative zero.

Parameters

- a* - an argument.
- b* - another argument.

Returns

the smaller of *a* and *b*.

6.432.3.17 `static short decaf::lang::Math::min (short a, short b) [inline, static]`

Returns the double value that is closest in value to the argument and is equal to a mathematical integer.

If two double values that are mathematical integers are equally close, the result is the integer value that is even. Special cases:

- o If the argument value is already equal to a mathematical integer, then the result is the same as the argument.
- o If the argument is NaN or an infinity or positive zero or negative zero, then the result is the same as the argument.

Parameters

- value* - the value to round to the nearest integer

Returns

the rounded value Returns the smaller of two short values. That is, the result is the argument closer to the value of `decaf::lang::Short::MIN_VALUE` (p. 2738). If the arguments have the same value, the result is that same value.

Parameters

- a* - an argument.
- b* - another argument.

Returns

the smaller of *a* and *b*.

6.432.3.18 `static double decaf::lang::Math::pow (double base, double exp)`
[static]

Returns the value of the first argument raised to the power of the second argument.

Special cases:

- o If the second argument is positive or negative zero, then the result is 1.0.
- o If the second argument is 1.0, then the result is the same as the first argument.
- o If the second argument is NaN, then the result is NaN.
- o If the first argument is NaN and the second argument is nonzero, then the result is NaN.

Parameters

- base* - the base
- exp* - the exponent

Returns

the base raised to the power of *exp*.

6.432.3.19 `static double decaf::lang::Math::random ()` [static]

Computes the remainder operation on two arguments as prescribed by the IEEE 754 standard.

The remainder value is mathematically equal to $f1 - f2 \times n$, where *n* is the mathematical integer closest to the exact mathematical value of the quotient $f1/f2$, and if two mathematical integers are equally close to $f1/f2$, then *n* is the integer that is even. If the remainder is zero, its sign is the same as the sign of the first argument. Special cases:

- o If either argument is NaN, or the first argument is infinite, or the second argument is positive zero or negative zero, then the result is NaN.
- o If the first argument is finite and the second argument is infinite, then the result is the same as the first argument.

Parameters

- f1* - the dividend.
- f2* - the divisor

Returns

the IEEE remainder of value Returns a double value with a positive sign, greater than or equal to 0.0 and less than 1.0. Returned values are chosen pseudorandomly with (approximately) uniform distribution from that range.

When this method is first called, it creates a single new pseudorandom-number generator; This new pseudorandom-number generator is used thereafter for all calls to this method and is used nowhere else.

This method is properly synchronized to allow correct use by more than one thread. However, if many threads need to generate pseudorandom numbers at a great rate, it may reduce contention for each thread to have its own pseudorandom-number generator.

Returns

a pseudorandom double greater than or equal to 0.0 and less than 1.0.

6.432.3.20 static long long decaf::lang::Math::round (double *value*) [static]

Returns the closest long long to the argument.

The result is rounded to an integer by adding 1/2, taking the floor of the result, and casting the result to type long long. In other words, the result is equal to the value of the expression: (long long)Math.floor (p. 2004)(a + 0.5d)

o If the argument is NaN, the result is 0. o If the argument is negative infinity or any value less than or equal to the value of **Long::MIN_VALUE** (p. 1947), the result is equal to the value of **Long::MIN_VALUE** (p. 1947). o If the argument is positive infinity or any value greater than or equal to the value of **Long::MAX_VALUE** (p. 1947), the result is equal to the value of **Long::MAX_VALUE** (p. 1947).

Parameters

value - the value to round

Returns

the value of the argument rounded to the nearest integral value.

6.432.3.21 static int decaf::lang::Math::round (float *value*) [static]

Returns the closest int to the argument.

The result is rounded to an integer by adding 1/2, taking the floor of the result, and casting the result to type int. In other words, the result is equal to the value of the expression: (int)Math.floor (p. 2004)(a + 0.5f)

o If the argument is NaN, the result is 0. o If the argument is negative infinity or any value less than or equal to the value of **Integer::MIN_VALUE** (p. 1666), the result is equal to the value of **Integer::MIN_VALUE** (p. 1666). o If the argument is positive infinity or any value greater than or equal to the value of **Integer::MAX_VALUE** (p. 1666), the result is equal to the value of **Integer::MAX_VALUE** (p. 1666).

Parameters

value - the value to round

Returns

the value of the argument rounded to the nearest integral value.

6.432.3.22 static double decaf::lang::Math::signum (double *value*) [static]

Returns the signum function of the argument; zero if the argument is zero, 1.0f if the argument is greater than zero, -1.0f if the argument is less than zero.

Special Cases:

o If the argument is NaN, then the result is NaN. o If the argument is positive zero or negative zero, then the result is the same as the argument.

Parameters

value - the floating-point value whose signum is to be returned

Returns

the signum function of the argument

6.432.3.23 static float decaf::lang::Math::signum (float *value*) [static]

Returns Euler's number e raised to the power of a double value.

Special cases:

o If the argument is NaN, the result is NaN. o If the argument is positive infinity, then the result is positive infinity. o If the argument is negative infinity, then the result is positive zero.

Parameters

value - the exponent to raise e to

Returns

the value e^x , where e is the base of the natural logarithms. Returns $e^x - 1$. Note that for values of x near 0, the exact sum of $\expm1(x) + 1$ is much closer to the true result of e^x than $\exp(x)$. Special cases:

o If the argument is NaN, the result is NaN. o If the argument is positive infinity, then the result is positive infinity. o If the argument is negative infinity, then the result is -1.0. o If the argument is zero, then the result is a zero with the same sign as the argument.

Parameters

value - the value to raise $e^x - 1$

Returns

the value $e^x - 1$. Returns $\sqrt{x^2 + y^2}$ without intermediate overflow or underflow. Special cases:

If either argument is infinite, then the result is positive infinity. If either argument is NaN and neither argument is infinite, then the result is NaN.

Parameters

x - an argument

y - another argument

Returns

the $\sqrt{x^2 + y^2}$ without intermediate overflow or underflow Returns the signum function of the argument; zero if the argument is zero, 1.0 if the argument is greater than zero, -1.0 if the argument is less than zero. Special Cases:

- o If the argument is NaN, then the result is NaN.
- o If the argument is positive zero or negative zero, then the result is the same as the argument.

Parameters

value - the floating-point value whose signum is to be returned

Returns

the signum function of the argument

6.432.3.24 static double decaf::lang::Math::sqrt (double *value*) [static]

Returns the arc cosine of an angle, in the range of 0.0 through pi.

Special case:

- o If the argument is NaN or its absolute value is greater than 1, then the result is NaN.

Parameters

value - the value to return the arc cosine of.

Returns

arc cosine of value in radians. Returns the arc sine of an angle, in the range of -pi/2 through pi/2. Special cases:

- o If the argument is NaN or its absolute value is greater than 1, then the result is NaN.
- o If the argument is zero, then the result is a zero with the same sign as the argument.

Parameters

value - the value to return the arc cosine of.

Returns

arc cosine of value in radians. Returns the arc tangent of an angle, in the range of -pi/2 through pi/2. Special cases:

- o If the argument is NaN, then the result is NaN.
- o If the argument is zero, then the result is a zero with the same sign as the argument.

Parameters

value - the value to return the arc cosine of.

Returns

arc tangent of value in radians. Converts rectangular coordinates (x, y) to polar (r, theta). This method computes the phase theta by computing an arc tangent of y/x in the range of -pi to pi. Special cases:

o If either argument is NaN, then the result is NaN. o If the first argument is positive zero and the second argument is positive, or the first argument is positive and finite and the second argument is positive infinity, then the result is positive zero. o If the first argument is negative zero and the second argument is positive, or the first argument is negative and finite and the second argument is positive infinity, then the result is negative zero. o If the first argument is positive zero and the second argument is negative, or the first argument is positive and finite and the second argument is negative infinity, then the result is the double value closest to pi. o If the first argument is negative zero and the second argument is negative, or the first argument is negative and finite and the second argument is negative infinity, then the result is the double value closest to -pi. o If the first argument is positive and the second argument is positive zero or negative zero, or the first argument is positive infinity and the second argument is finite, then the result is the double value closest to pi/2. o If the first argument is negative and the second argument is positive zero or negative zero, or the first argument is negative infinity and the second argument is finite, then the result is the double value closest to -pi/2. o If both arguments are positive infinity, then the result is the double value closest to pi/4. o If the first argument is positive infinity and the second argument is negative infinity, then the result is the double value closest to 3*pi/4. o If the first argument is negative infinity and the second argument is positive infinity, then the result is the double value closest to -pi/4. o If both arguments are negative infinity, then the result is the double value closest to -3*pi/4.

Parameters

y - the ordinate coordinate
x - the abscissa coordinate

Returns

the theta component of the point (r, theta) in polar coordinates that corresponds to the point (x, y) in Cartesian coordinates. Returns the cube root of a double value. For positive finite x, `cbrt(-x) == -cbrt(x)`; that is, the cube root of a negative value is the negative of the cube root of that value's magnitude. Special cases:

o If the argument is NaN, then the result is NaN. o If the argument is infinite, then the result is an infinity with the same sign as the argument. o If the argument is zero, then the result is a zero with the same sign as the argument.

Parameters

value - the double to compute the cube root of

Returns

the cube root of value Returns the trigonometric cosine of an angle. Special cases:

o If the argument is NaN or an infinity, then the result is NaN.

Parameters

value - an value in radians

Returns

the cosine of the argument. Returns the hyperbolic cosine of a double value. The hyperbolic cosine of x is defined to be $(e^x + e^{-x})/2$ where e is Euler's number. Special cases:

o If the argument is NaN, then the result is NaN. o If the argument is infinite, then the result is positive infinity. o If the argument is zero, then the result is 1.0.

Parameters

value - the number whose hyperbolic cosine is to be found

Returns

the hyperbolic cosine of value Returns the trigonometric sine of an angle. Special case:

o If the argument is NaN or an infinity, then the result is NaN. o If the argument is zero, then the result is a zero with the same sign as the argument.

Parameters

value - the number whose sin is to be found

Returns

the sine of value Returns the hyperbolic sine of a double value. The hyperbolic sine of x is defined to be $(e^x - e^{-x})/2$ where e is Euler's number. Special cases:

o If the argument is NaN, then the result is NaN. o If the argument is infinite, then the result is an infinity with the same sign as the argument. o If the argument is zero, then the result is a zero with the same sign as the argument.

Parameters

value - the number whose hyperbolic sin is to be found

Returns

the hyperbolic sine of value Returns the trigonometric tangent of an angle. Special cases:

o If the argument is NaN or an infinity, then the result is NaN. o If the argument is zero, then the result is a zero with the same sign as the argument.

Parameters

value - the number whose tangent is to be found

Returns

the tangent of value Returns the hyperbolic tangent of a double value. The hyperbolic tangent of x is defined to be $(e^x - e^{-x})/(e^x + e^{-x})$, in other words, $\sinh(x)/\cosh(x)$. Note that the absolute value of the exact tanh is always less than 1. Special cases:

o If the argument is NaN, then the result is NaN. o If the argument is zero, then the result is a zero with the same sign as the argument. o If the argument is positive infinity, then the result is +1.0. o If the argument is negative infinity, then the result is -1.0.

Parameters

value - the number whose hyperbolic tangent is to be found

Returns

the hyperbolic cosine of value Returns the correctly rounded positive square root of a double value. Special cases:

o If the argument is NaN or less than zero, then the result is NaN. o If the argument is positive infinity, then the result is positive infinity. o If the argument is positive zero or negative zero, then the result is the same as the argument.

Otherwise, the result is the double value closest to the true mathematical square root of the argument value.

Parameters

value - the value to find the square root of
the square root of the argument.

6.432.3.25 `static double decaf::lang::Math::toDegrees (double angrad)` [inline, static]

Returns the measure in degrees of the supplied radian angle.

Parameters

angrad - an angle in radians

Returns

the degree measure of the angle.

6.432.3.26 `static double decaf::lang::Math::toRadians (double angdeg)` [inline, static]

Returns the measure in radians of the supplied degree angle.

Parameters

angdeg - an angle in degrees

Returns

the radian measure of the angle.

6.432.4 Field Documentation

6.432.4.1 `const double decaf::lang::Math::E` [static]

6.432.4.2 `const double decaf::lang::Math::PI` [static]

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Math.h`

6.433 activemq::util::MemoryUsage Class Reference

```
#include <src/main/activemq/util/MemoryUsage.h>
```

Inheritance diagram for `activemq::util::MemoryUsage`:

Public Member Functions

- **MemoryUsage** ()
Default Constructor.
- **MemoryUsage** (unsigned long long limit)
*Creates an instance of an **Usage** (p. 3137) monitor with a set limit.*
- virtual **~MemoryUsage** ()
- virtual void **waitForSpace** ()
*Waits forever for more space to be returned to this **Usage** (p. 3137) Manager.*
- virtual void **waitForSpace** (unsigned int timeout)
*Waits for more space to be returned to this **Usage** (p. 3137) Manager, times out when the given time span in milliseconds elapses.*
- virtual void **enqueueUsage** (unsigned long long value)
Tries to increase the usage by value amount but blocks if this object is currently full.
- virtual void **increaseUsage** (unsigned long long value)
Increases the usage by the value amount.
- virtual void **decreaseUsage** (unsigned long long value)
Decreases the usage by the value amount.
- virtual bool **isFull** () const
*Returns true if this **Usage** (p. 3137) instance is full, i.e.*
- unsigned long long **getUsage** () const
Gets the current usage amount.
- void **setUsage** (unsigned long long usage)
Sets the current usage amount.
- unsigned long long **getLimit** () const
Gets the current limit amount.
- void **setLimit** (unsigned long long limit)
Sets the current limit amount.

6.433.1 Constructor & Destructor Documentation

6.433.1.1 activemq::util::MemoryUsage::MemoryUsage ()

Default Constructor.

6.433.1.2 `activemq::util::MemoryUsage::MemoryUsage (unsigned long long limit)`

Creates an instance of an `Usage` (p. 3137) monitor with a set limit.

Parameters

limit - amount of memory this manager allows.

6.433.1.3 `virtual activemq::util::MemoryUsage::~~MemoryUsage () [virtual]`

6.433.2 Member Function Documentation

6.433.2.1 `virtual void activemq::util::MemoryUsage::decreaseUsage (unsigned long long value) [virtual]`

Decreases the usage by the value amount.

Parameters

value Amount of space to return to the pool

Implements `activemq::util::Usage` (p. 3137).

6.433.2.2 `virtual void activemq::util::MemoryUsage::enqueueUsage (unsigned long long value) [inline, virtual]`

Tries to increase the usage by value amount but blocks if this object is currently full.

Parameters

value Amount of usage in bytes to add.

Implements `activemq::util::Usage` (p. 3138).

6.433.2.3 `unsigned long long activemq::util::MemoryUsage::getLimit () const [inline]`

Gets the current limit amount.

Returns

the amount that can be used before full.

6.433.2.4 `unsigned long long activemq::util::MemoryUsage::getUsage () const [inline]`

Gets the current usage amount.

Returns

the amount of bytes currently used.

6.433.2.5 `virtual void activemq::util::MemoryUsage::increaseUsage (unsigned long long value) [virtual]`

Increases the usage by the value amount.

Parameters

value Amount of usage to add.

Implements `activemq::util::Usage` (p. 3138).

6.433.2.6 `virtual bool activemq::util::MemoryUsage::isFull () const [virtual]`

Returns true if this `Usage` (p. 3137) instance is full, i.e.

`Usage` (p. 3137) $\geq 100\%$

Implements `activemq::util::Usage` (p. 3138).

6.433.2.7 `void activemq::util::MemoryUsage::setLimit (unsigned long long limit) [inline]`

Sets the current limit amount.

Parameters

limit - The amount that can be used before full.

6.433.2.8 `void activemq::util::MemoryUsage::setUsage (unsigned long long usage) [inline]`

Sets the current usage amount.

Parameters

usage - The amount to tag as used.

6.433.2.9 `virtual void activemq::util::MemoryUsage::waitForSpace () [virtual]`

Waits forever for more space to be returned to this `Usage` (p. 3137) Manager.

Implements `activemq::util::Usage` (p. 3138).

6.433.2.10 `virtual void activemq::util::MemoryUsage::waitForSpace (unsigned int timeout) [virtual]`

Waits for more space to be returned to this `Usage` (p. 3137) Manager, times out when the given time span in milliseconds elapses.

Parameters

timeout The time to wait for more space.

Implements **activemq::util::Usage** (p. 3138).

The documentation for this class was generated from the following file:

- `src/main/activemq/util/MemoryUsage.h`

6.434 activemq::commands::Message Class Reference

```
#include <src/main/activemq/commands/Message.h>
```

Inheritance diagram for `activemq::commands::Message`:

Public Member Functions

- **Message** ()
- virtual **~Message** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **Message * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1372) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent.*
- virtual void **beforeMarshal** (**wireformat::WireFormat** *wireFormat AMQCPP_UNUSED) throw (decaf::io::IOException)
Handles the marshaling of the objects properties into the internal byte array before the object is marshaled to the wire.
- virtual void **afterUnmarshal** (**wireformat::WireFormat** *wireFormat AMQCPP_UNUSED) throw (decaf::io::IOException)
Called after unmarshaling is started to cleanup the object being unmarshaled.
- virtual bool **isMarshalAware** () const
Indicates that this command is aware of Marshaling, and needs to have its Marshaling methods invoked.
- virtual void **setAckHandler** (const **Pointer**< **core::ActiveMQAckHandler** > &handler)

*Sets the Acknowledgment Handler that this **Message** (p. 2018) will use when the Acknowledge method is called.*

- virtual **Pointer**< **core::ActiveMQAckHandler** > **getAckHandler** () const
*Gets the Acknowledgment Handler that this **Message** (p. 2018) will use when the Acknowledge method is called.*
- virtual unsigned int **getSize** () const
Returns the Size of this message in Bytes.
- virtual bool **isExpired** () const
Returns if this message has expired, meaning that its Expiration time has elapsed.
- virtual void **onSend** ()
*Allows derived **Message** (p. 2018) classes to perform tasks before a message is sent.*
- **util::PrimitiveMap** & **getMessageProperties** ()
Gets a reference to the Message's Properties object, allows the derived classes to get and set their own specific properties.
- const **util::PrimitiveMap** & **getMessageProperties** () const
- bool **isReadOnlyProperties** () const
*Returns if the **Message** (p. 2018) Properties Are Read Only.*
- void **setReadOnlyProperties** (bool value)
*Set the Read Only State of the **Message** (p. 2018) Properties.*
- bool **isReadOnlyBody** () const
*Returns if the **Message** (p. 2018) Body is Read Only.*
- void **setReadOnlyBody** (bool value)
*Set the Read Only State of the **Message** (p. 2018) Content.*
- virtual const **Pointer**< **ProducerId** > & **getProducerId** () const
- virtual **Pointer**< **ProducerId** > & **getProducerId** ()
- virtual void **setProducerId** (const **Pointer**< **ProducerId** > &producerId)
- virtual const **Pointer**< **ActiveMQDestination** > & **getDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &destination)
- virtual const **Pointer**< **TransactionId** > & **getTransactionId** () const
- virtual **Pointer**< **TransactionId** > & **getTransactionId** ()
- virtual void **setTransactionId** (const **Pointer**< **TransactionId** > &transactionId)
- virtual const **Pointer**< **ActiveMQDestination** > & **getOriginalDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getOriginalDestination** ()
- virtual void **setOriginalDestination** (const **Pointer**< **ActiveMQDestination** > &originalDestination)
- virtual const **Pointer**< **MessageId** > & **getMessageId** () const
- virtual **Pointer**< **MessageId** > & **getMessageId** ()
- virtual void **setMessageId** (const **Pointer**< **MessageId** > &messageId)
- virtual const **Pointer**< **TransactionId** > & **getOriginalTransactionId** () const
- virtual **Pointer**< **TransactionId** > & **getOriginalTransactionId** ()

- virtual void **setOriginalTransactionId** (const **Pointer**< **TransactionId** > &**originalTransactionId**)
- virtual const std::string & **getGroupID** () const
- virtual std::string & **getGroupID** ()
- virtual void **setGroupID** (const std::string &**groupID**)
- virtual int **getGroupSequence** () const
- virtual void **setGroupSequence** (int **groupSequence**)
- virtual const std::string & **getCorrelationId** () const
- virtual std::string & **getCorrelationId** ()
- virtual void **setCorrelationId** (const std::string &**correlationId**)
- virtual bool **isPersistent** () const
- virtual void **setPersistent** (bool **persistent**)
- virtual long long **getExpiration** () const
- virtual void **setExpiration** (long long **expiration**)
- virtual unsigned char **getPriority** () const
- virtual void **setPriority** (unsigned char **priority**)
- virtual const **Pointer**< **ActiveMQDestination** > & **getReplyTo** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getReplyTo** ()
- virtual void **setReplyTo** (const **Pointer**< **ActiveMQDestination** > &**replyTo**)
- virtual long long **getTimestamp** () const
- virtual void **setTimestamp** (long long **timestamp**)
- virtual const std::string & **getType** () const
- virtual std::string & **getType** ()
- virtual void **setType** (const std::string &**type**)
- virtual const std::vector< unsigned char > & **getContent** () const
- virtual std::vector< unsigned char > & **getContent** ()
- virtual void **setContent** (const std::vector< unsigned char > &**content**)
- virtual const std::vector< unsigned char > & **getMarshaledProperties** () const
- virtual std::vector< unsigned char > & **getMarshaledProperties** ()
- virtual void **setMarshaledProperties** (const std::vector< unsigned char > &**marshalledProperties**)
- virtual const **Pointer**< **DataStructure** > & **getDataStructure** () const
- virtual **Pointer**< **DataStructure** > & **getDataStructure** ()
- virtual void **setDataStructure** (const **Pointer**< **DataStructure** > &**dataStructure**)
- virtual const **Pointer**< **ConsumerId** > & **getTargetConsumerId** () const
- virtual **Pointer**< **ConsumerId** > & **getTargetConsumerId** ()
- virtual void **setTargetConsumerId** (const **Pointer**< **ConsumerId** > &**targetConsumerId**)
- virtual bool **isCompressed** () const
- virtual void **setCompressed** (bool **compressed**)
- virtual int **getRedeliveryCounter** () const
- virtual void **setRedeliveryCounter** (int **redeliveryCounter**)
- virtual const std::vector< **decaf::lang::Pointer**< **BrokerId** > > & **getBrokerPath** () const
- virtual std::vector< **decaf::lang::Pointer**< **BrokerId** > > & **getBrokerPath** ()
- virtual void **setBrokerPath** (const std::vector< **decaf::lang::Pointer**< **BrokerId** > > &**brokerPath**)
- virtual long long **getArrival** () const
- virtual void **setArrival** (long long **arrival**)
- virtual const std::string & **getUserID** () const

- virtual std::string & **getUserID** ()
- virtual void **setUserID** (const std::string &userID)
- virtual bool **isRecievedByDFBridge** () const
- virtual void **setRecievedByDFBridge** (bool recievedByDFBridge)
- virtual bool **isDroppable** () const
- virtual void **setDroppable** (bool droppable)
- virtual const std::vector< **decaf::lang::Pointer**< **BrokerId** > > & **getCluster** () const
- virtual std::vector< **decaf::lang::Pointer**< **BrokerId** > > & **getCluster** ()
- virtual void **setCluster** (const std::vector< **decaf::lang::Pointer**< **BrokerId** > > &cluster)
- virtual long long **getBrokerInTime** () const
- virtual void **setBrokerInTime** (long long brokerInTime)
- virtual long long **getBrokerOutTime** () const
- virtual void **setBrokerOutTime** (long long brokerOutTime)
- virtual bool **isMessage** () const
- virtual **Pointer**< **Command** > **visit** (**activemq::state::CommandVisitor** *visitor) throw (exceptions::ActiveMQException)

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_MESSAGE** = 0

Protected Member Functions

- **Message** (const **Message** &)
- **Message** & **operator=** (const **Message** &)

Protected Attributes

- **Pointer**< **ProducerId** > **producerId**
- **Pointer**< **ActiveMQDestination** > **destination**
- **Pointer**< **TransactionId** > **transactionId**
- **Pointer**< **ActiveMQDestination** > **originalDestination**
- **Pointer**< **MessageId** > **messageId**
- **Pointer**< **TransactionId** > **originalTransactionId**
- std::string **groupID**
- int **groupSequence**
- std::string **correlationId**
- bool **persistent**
- long long **expiration**
- unsigned char **priority**
- **Pointer**< **ActiveMQDestination** > **replyTo**
- long long **timestamp**
- std::string **type**
- std::vector< unsigned char > **content**
- std::vector< unsigned char > **marshalledProperties**

- **Pointer< DataStructure > dataStructure**
- **Pointer< ConsumerId > targetConsumerId**
- **bool compressed**
- **int redeliveryCounter**
- **std::vector< decaf::lang::Pointer< BrokerId > > brokerPath**
- **long long arrival**
- **std::string userID**
- **bool recievedByDFBridge**
- **bool droppable**
- **std::vector< decaf::lang::Pointer< BrokerId > > cluster**
- **long long brokerInTime**
- **long long brokerOutTime**

Static Protected Attributes

- **static const unsigned int DEFAULT_MESSAGE_SIZE = 1024**

6.434.1 Constructor & Destructor Documentation

6.434.1.1 **activemq::commands::Message::Message (const Message &)** [inline, protected]

6.434.1.2 **activemq::commands::Message::Message ()**

6.434.1.3 **virtual activemq::commands::Message::~Message ()** [virtual]

6.434.2 Member Function Documentation

6.434.2.1 **virtual void activemq::commands::Message::afterUnmarshal (wireformat::WireFormat *wireFormat *AMQCPP_UNUSED*) throw (decaf::io::IOException)** [virtual]

Called after unmarshaling is started to cleanup the object being unmarshaled.

Parameters

wireFormat - the wireformat object to control unmarshaling

Reimplemented from **activemq::commands::BaseDataStructure** (p.660).

6.434.2.2 **virtual void activemq::commands::Message::beforeMarshal (wireformat::WireFormat *wireFormat *AMQCPP_UNUSED*) throw (decaf::io::IOException)** [virtual]

Handles the marshaling of the objects properties into the internal byte array before the object is marshaled to the wire.

Parameters

wireFormat - the wireformat controller

Reimplemented from **activemq::commands::BaseDataStructure** (p.660).

6.434.2.3 `virtual Message* activemq::commands::Message::cloneDataStructure () const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1373).

Reimplemented in `activemq::commands::ActiveMQBlobMessage` (p. 143), `activemq::commands::ActiveMQBytesMessage` (p. 170), `activemq::commands::ActiveMQMapMessage` (p. 276), `activemq::commands::ActiveMQMessage` (p. 306), `activemq::commands::ActiveMQObjectMessage` (p. 345), `activemq::commands::ActiveMQStreamMessage` (p. 425), and `activemq::commands::ActiveMQTextMessage` (p. 527).

6.434.2.4 `virtual void activemq::commands::Message::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

Reimplemented from `activemq::commands::BaseCommand` (p. 598).

Reimplemented in `activemq::commands::ActiveMQBlobMessage` (p. 143), `activemq::commands::ActiveMQBytesMessage` (p. 170), `activemq::commands::ActiveMQMapMessage` (p. 276), `activemq::commands::ActiveMQMessage` (p. 306), `activemq::commands::ActiveMQObjectMessage` (p. 346), `activemq::commands::ActiveMQStreamMessage` (p. 425), and `activemq::commands::ActiveMQTextMessage` (p. 527).

6.434.2.5 `virtual bool activemq::commands::Message::equals (const DataStructure * value) const [virtual]`

Compares the `DataStructure` (p. 1372) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if `DataStructure`'s are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 598).

Reimplemented in `activemq::commands::ActiveMQBlobMessage` (p. 144), `activemq::commands::ActiveMQBytesMessage` (p. 171), `activemq::commands::ActiveMQMapMessage` (p. 276), `activemq::commands::ActiveMQMessage` (p. 306), `activemq::commands::ActiveMQMessageTemplate< T >` (p. 331), and `activemq::commands::ActiveMQObjectMessage`

(p. 346), `activemq::commands::ActiveMQStreamMessage` (p. 426), `activemq::commands::ActiveMQTextMessage` (p. 527), `activemq::commands::ActiveMQMessageTemplate<cms::BytesMessage>` (p. 331), `activemq::commands::ActiveMQMessageTemplate<cms::MapMessage>` (p. 331), `activemq::commands::ActiveMQMessageTemplate<cms::Message>` (p. 331), `activemq::commands::ActiveMQMessageTemplate<cms::StreamMessage>` (p. 331), `activemq::commands::ActiveMQMessageTemplate<cms::TextMessage>` (p. 331), and `activemq::commands::ActiveMQMessageTemplate<cms::ObjectMessage>` (p. 331).

6.434.2.6 `virtual Pointer<core::ActiveMQAckHandler>
activemq::commands::Message::getAckHandler () const [inline,
virtual]`

Gets the Acknowledgment Handler that this **Message** (p. 2018) will use when the Acknowledge method is called.

Returns

handler `ActiveMQAckHandler` to call or `NULL` if not set

- 6.434.2.7 virtual long long activemq::commands::Message::getArrival () const [virtual]
- 6.434.2.8 virtual long long activemq::commands::Message::getBrokerInTime () const [virtual]
- 6.434.2.9 virtual long long activemq::commands::Message::getBrokerOutTime () const [virtual]
- 6.434.2.10 virtual const std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::Message::getBrokerPath () const [virtual]
- 6.434.2.11 virtual std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::Message::getBrokerPath () [virtual]
- 6.434.2.12 virtual std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::Message::getCluster () [virtual]
- 6.434.2.13 virtual const std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::Message::getCluster () const [virtual]
- 6.434.2.14 virtual const std::vector<unsigned char>& activemq::commands::Message::getContent () const [virtual]
- 6.434.2.15 virtual std::vector<unsigned char>& activemq::commands::Message::getContent () [virtual]
- 6.434.2.16 virtual std::string& activemq::commands::Message::getCorrelationId () [virtual]
- 6.434.2.17 virtual const std::string& activemq::commands::Message::getCorrelationId () const [virtual]
- 6.434.2.18 virtual const Pointer<DataStructure>& activemq::commands::Message::getDataStructure () const [virtual]
- 6.434.2.19 virtual Pointer<DataStructure>& activemq::commands::Message::getDataStructure () [virtual]
- 6.434.2.20 virtual unsigned char activemq::commands::Message::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataStructure** (p. 1372) type copy.

Implements **activemq::commands::DataStructure** (p. 1375).

Reimplemented in **activemq::commands::ActiveMQBlobMessage** (p. 144), **activemq::commands::ActiveMQBytesMessage** (p. 172), **activemq::commands::ActiveMQMapMessage** (p. 278), **activemq::commands::ActiveMQMessage** (p. 306), **activemq::commands::ActiveMQObjectMessage** (p. 346), **activemq::commands::ActiveMQStreamMessage** (p. 426), and **activemq::commands::ActiveMQTextMessage** (p. 528).

- 6.434.2.21 `virtual const Pointer<ActiveMQDestination>& activemq::commands::Message::getDestination () const [virtual]`
- 6.434.2.22 `virtual Pointer<ActiveMQDestination>& activemq::commands::Message::getDestination () [virtual]`
- 6.434.2.23 `virtual long long activemq::commands::Message::getExpiration () const [virtual]`
- 6.434.2.24 `virtual const std::string& activemq::commands::Message::getGroupID () const [virtual]`
- 6.434.2.25 `virtual std::string& activemq::commands::Message::getGroupID () [virtual]`
- 6.434.2.26 `virtual int activemq::commands::Message::getGroupSequence () const [virtual]`
- 6.434.2.27 `virtual const std::vector<unsigned char>& activemq::commands::Message::getMarshaledProperties () const [virtual]`
- 6.434.2.28 `virtual std::vector<unsigned char>& activemq::commands::Message::getMarshaledProperties () [virtual]`
- 6.434.2.29 `virtual const Pointer<MessageId>& activemq::commands::Message::getMessageId () const [virtual]`
- 6.434.2.30 `virtual Pointer<MessageId>& activemq::commands::Message::getMessageId () [virtual]`
- 6.434.2.31 `const util::PrimitiveMap& activemq::commands::Message::getMessageProperties () const [inline]`
- 6.434.2.32 `util::PrimitiveMap& activemq::commands::Message::getMessageProperties () [inline]`

Gets a reference to the Message's Properties object, allows the derived classes to get and set their own specific properties.

Returns

a reference to the Primitive Map that holds message properties.

- 6.434.2.33** `virtual const Pointer<ActiveMQDestination>& activemq::commands::Message::getOriginalDestination () const [virtual]`
- 6.434.2.34** `virtual Pointer<ActiveMQDestination>& activemq::commands::Message::getOriginalDestination () [virtual]`
- 6.434.2.35** `virtual Pointer<TransactionId>& activemq::commands::Message::getOriginalTransactionId () [virtual]`
- 6.434.2.36** `virtual const Pointer<TransactionId>& activemq::commands::Message::getOriginalTransactionId () const [virtual]`
- 6.434.2.37** `virtual unsigned char activemq::commands::Message::getPriority () const [virtual]`
- 6.434.2.38** `virtual Pointer<ProducerId>& activemq::commands::Message::getProducerId () [virtual]`
- 6.434.2.39** `virtual const Pointer<ProducerId>& activemq::commands::Message::getProducerId () const [virtual]`
- 6.434.2.40** `virtual int activemq::commands::Message::getRedeliveryCounter () const [virtual]`
- 6.434.2.41** `virtual const Pointer<ActiveMQDestination>& activemq::commands::Message::getReplyTo () const [virtual]`
- 6.434.2.42** `virtual Pointer<ActiveMQDestination>& activemq::commands::Message::getReplyTo () [virtual]`
- 6.434.2.43** `virtual unsigned int activemq::commands::Message::getSize () const [virtual]`

Returns the Size of this message in Bytes.

Returns

number of bytes this message equates to.

Reimplemented in `activemq::commands::ActiveMQTextMessage` (p. 528).

- 6.434.2.44 `virtual const Pointer<ConsumerId>& activemq::commands::Message::getTargetConsumerId () const [virtual]`
- 6.434.2.45 `virtual Pointer<ConsumerId>& activemq::commands::Message::getTargetConsumerId () [virtual]`
- 6.434.2.46 `virtual long long activemq::commands::Message::getTimestamp () const [virtual]`
- 6.434.2.47 `virtual const Pointer<TransactionId>& activemq::commands::Message::getTransactionId () const [virtual]`
- 6.434.2.48 `virtual Pointer<TransactionId>& activemq::commands::Message::getTransactionId () [virtual]`
- 6.434.2.49 `virtual std::string& activemq::commands::Message::getType () [virtual]`
- 6.434.2.50 `virtual const std::string& activemq::commands::Message::getType () const [virtual]`
- 6.434.2.51 `virtual const std::string& activemq::commands::Message::getUserID () const [virtual]`
- 6.434.2.52 `virtual std::string& activemq::commands::Message::getUserID () [virtual]`
- 6.434.2.53 `virtual bool activemq::commands::Message::isCompressed () const [virtual]`
- 6.434.2.54 `virtual bool activemq::commands::Message::isDroppable () const [virtual]`
- 6.434.2.55 `virtual bool activemq::commands::Message::isExpired () const [virtual]`

Returns if this message has expired, meaning that its Expiration time has elapsed.

Returns

true if message is expired.

- 6.434.2.56 `virtual bool activemq::commands::Message::isMarshalAware () const [inline, virtual]`

Indicates that this command is aware of Marshaling, and needs to have its Marshaling methods invoked.

Returns

boolean indicating desire to be in marshaling stages

Reimplemented from `activemq::commands::BaseDataStructure` (p. 662).

Reimplemented in `activemq::commands::ActiveMQMapMessage` (p. 280).

6.434.2.57 `virtual bool activemq::commands::Message::isMessage () const`
[inline, virtual]

Returns

an answer of true to the `isMessage()` (p. 2030) query.

Reimplemented from `activemq::commands::BaseCommand` (p. 600).

6.434.2.58 `virtual bool activemq::commands::Message::isPersistent () const`
[virtual]

6.434.2.59 `bool activemq::commands::Message::isReadOnlyBody () const`
[inline]

Returns if the `Message` (p. 2018) Body is Read Only.

Returns

true if `Message` (p. 2018) Content is Read Only.

6.434.2.60 `bool activemq::commands::Message::isReadOnlyProperties () const`
[inline]

Returns if the `Message` (p. 2018) Properties Are Read Only.

Returns

true if `Message` (p. 2018) Properties are Read Only.

6.434.2.61 `virtual bool activemq::commands::Message::isRecievedByDFBridge () const` [virtual]

6.434.2.62 `virtual void activemq::commands::Message::onSend ()` [inline, virtual]

Allows derived `Message` (p. 2018) classes to perform tasks before a message is sent.

Reimplemented in `activemq::commands::ActiveMQBytesMessage` (p. 172),
`activemq::commands::ActiveMQMessageTemplate< T >` (p. 338),
`activemq::commands::ActiveMQStreamMessage` (p. 426),
`activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >` (p. 338),
`activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >` (p. 338),
`activemq::commands::ActiveMQMessageTemplate< cms::Message >` (p. 338),
`activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >` (p. 338),
`activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >` (p. 338), and
`activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >` (p. 338).

6.434.2.63 `Message& activemq::commands::Message::operator= (const Message
&) [inline, protected]`

6.434.2.64 `virtual void activemq::commands::Message::setAckHandler (const
Pointer< core::ActiveMQAckHandler > & handler) [inline,
virtual]`

Sets the Acknowledgment Handler that this **Message** (p.2018) will use when the Acknowledge method is called.

Parameters

handler ActiveMQAckHandler to call

- 6.434.2.65 virtual void activemq::commands::Message::setArrival (long long *arrival*) [virtual]
- 6.434.2.66 virtual void activemq::commands::Message::setBrokerInTime (long long *brokerInTime*) [virtual]
- 6.434.2.67 virtual void activemq::commands::Message::setBrokerOutTime (long long *brokerOutTime*) [virtual]
- 6.434.2.68 virtual void activemq::commands::Message::setBrokerPath (const std::vector< decaf::lang::Pointer< BrokerId > > & *brokerPath*) [virtual]
- 6.434.2.69 virtual void activemq::commands::Message::setCluster (const std::vector< decaf::lang::Pointer< BrokerId > > & *cluster*) [virtual]
- 6.434.2.70 virtual void activemq::commands::Message::setCompressed (bool *compressed*) [virtual]
- 6.434.2.71 virtual void activemq::commands::Message::setContent (const std::vector< unsigned char > & *content*) [virtual]
- 6.434.2.72 virtual void activemq::commands::Message::setCorrelationId (const std::string & *correlationId*) [virtual]
- 6.434.2.73 virtual void activemq::commands::Message::setDataStructure (const Pointer< DataStructure > & *dataStructure*) [virtual]
- 6.434.2.74 virtual void activemq::commands::Message::setDestination (const Pointer< ActiveMQDestination > & *destination*) [virtual]
- 6.434.2.75 virtual void activemq::commands::Message::setDroppable (bool *droppable*) [virtual]
- 6.434.2.76 virtual void activemq::commands::Message::setExpiration (long long *expiration*) [virtual]
- 6.434.2.77 virtual void activemq::commands::Message::setGroupID (const std::string & *groupID*) [virtual]
- 6.434.2.78 virtual void activemq::commands::Message::setGroupSequence (int *groupSequence*) [virtual]
- 6.434.2.79 virtual void activemq::commands::Message::setMarshaledProperties (const std::vector< unsigned char > & *marshalledProperties*) [virtual]
- 6.434.2.80 virtual void activemq::commands::Message::setMessageId (const Pointer< MessageId > & *messageId*) [virtual]
- 6.434.2.81 virtual void activemq::commands::Message::setOriginalDestination (const Pointer< ActiveMQDestination > & *originalDestination*) [virtual]
- 6.434.2.82 virtual void activemq::commands::Message::setOriginalTransactionId (const Pointer< TransactionId > & *originalTransactionId*) [virtual]
- 6.434.2.83 virtual void activemq::commands::Message::setPersistent (bool *persistent*) [virtual]
- 6.434.2.84 virtual void activemq::commands::Message::setPriority (unsigned char *priority*) [virtual]

Parameters

value - true if Content should be read only.

6.434.2.87 `void activemq::commands::Message::setReadOnlyProperties (bool value) [inline]`

Set the Read Only State of the **Message** (p.2018) Properties.

Parameters

value - true if Properties should be read only.

6.434.2.88 `virtual void activemq::commands::Message::setRecievedByDFBridge (bool recievedByDFBridge) [virtual]`

6.434.2.89 `virtual void activemq::commands::Message::setRedeliveryCounter (int redeliveryCounter) [virtual]`

6.434.2.90 `virtual void activemq::commands::Message::setReplyTo (const Pointer< ActiveMQDestination > & replyTo) [virtual]`

6.434.2.91 `virtual void activemq::commands::Message::setTargetConsumerId (const Pointer< ConsumerId > & targetConsumerId) [virtual]`

6.434.2.92 `virtual void activemq::commands::Message::setTimestamp (long long timestamp) [virtual]`

6.434.2.93 `virtual void activemq::commands::Message::setTransactionId (const Pointer< TransactionId > & transactionId) [virtual]`

6.434.2.94 `virtual void activemq::commands::Message::setType (const std::string & type) [virtual]`

6.434.2.95 `virtual void activemq::commands::Message::setUserID (const std::string & userID) [virtual]`

6.434.2.96 `virtual std::string activemq::commands::Message::toString () const [virtual]`

Returns a string containing the information for this **DataStructure** (p.1372) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p.602).

Reimplemented in **activemq::commands::ActiveMQBlobMessage** (p.146), **activemq::commands::ActiveMQBytesMessage** (p.177), **activemq::commands::ActiveMQMapMessage** (p.284), **activemq::commands::ActiveMQMessage** (p.307), **activemq::commands::ActiveMQObjectMessage**

(p. 346), `activemq::commands::ActiveMQStreamMessage` (p. 432), and `activemq::commands::ActiveMQTextMessage` (p. 529).

```
6.434.2.97 virtual Pointer<Command> activemq::commands::Message::visit  
          ( activemq::state::CommandVisitor * visitor ) throw (   
            exceptions::ActiveMQException ) [virtual]
```

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

Returns

a **Response** (p. 2611) to the visitor being called or NULL if no response.

Implements `activemq::commands::Command` (p. 995).

6.434.3 Field Documentation

- 6.434.3.1 `long long activemq::commands::Message::arrival` [protected]
- 6.434.3.2 `long long activemq::commands::Message::brokerInTime` [protected]
- 6.434.3.3 `long long activemq::commands::Message::brokerOutTime` [protected]
- 6.434.3.4 `std::vector< decaf::lang::Pointer<BrokerId> >`
`activemq::commands::Message::brokerPath` [protected]
- 6.434.3.5 `std::vector< decaf::lang::Pointer<BrokerId> >`
`activemq::commands::Message::cluster` [protected]
- 6.434.3.6 `bool activemq::commands::Message::compressed` [protected]
- 6.434.3.7 `std::vector<unsigned char> activemq::commands::Message::content`
[protected]
- 6.434.3.8 `std::string activemq::commands::Message::correlationId` [protected]
- 6.434.3.9 `Pointer<DataStructure> activemq::commands::Message::dataStructure`
[protected]
- 6.434.3.10 `const unsigned int activemq::commands::Message::DEFAULT_-`
`MESSAGE_SIZE = 1024` [static, protected]
- 6.434.3.11 `Pointer<ActiveMQDestination> ac-`
`tivemq::commands::Message::destination` [protected]
- 6.434.3.12 `bool activemq::commands::Message::droppable` [protected]
- 6.434.3.13 `long long activemq::commands::Message::expiration` [protected]
- 6.434.3.14 `std::string activemq::commands::Message::groupId` [protected]
- 6.434.3.15 `int activemq::commands::Message::groupSequence` [protected]
- 6.434.3.16 `const unsigned char activemq::commands::Message::ID_MESSAGE = 0`
[static]
- 6.434.3.17 `std::vector<unsigned char> ac-`
`tivemq::commands::Message::marshalledProperties`
[protected]
- 6.434.3.18 `Pointer<MessageId> activemq::commands::Message::messageId`
[protected]
- 6.434.3.19 `Pointer<ActiveMQDestination> ac-`
`tivemq::commands::Message::originalDestination`
[protected]
- 6.434.3.20 `Pointer<TransactionId> ac-`
`tivemq::commands::Message::originalTransactionId`
[protected]

Generated on Mon Jan 24 2011 16:15:22 for activemq-cpp-3.1.0 by Doxygen

- 6.434.3.21 `bool activemq::commands::Message::persistent` [protected]
- 6.434.3.22 `unsigned char activemq::commands::Message::priority` [protected]
- 6.434.3.23 `Pointer<ProducerId> activemq::commands::Message::producerId`
[protected]

- src/main/activemq/commands/Message.h

6.435 cms::Message Class Reference

Root of all messages.

```
#include <src/main/cms/Message.h>
```

Inheritance diagram for cms::Message:

Public Member Functions

- virtual **~Message** ()
- virtual **Message * clone** () const =0
Clone this message exactly, returns a new instance that the caller is required to delete.
- virtual void **acknowledge** () const =0 throw (IllegalStateException, CMSEException)
Acknowledges all consumed messages of the session of this consumed message.
- virtual void **clearBody** ()=0 throw (CMSEException)
Clears out the body of the message.
- virtual void **clearProperties** ()=0 throw (CMSEException)
Clears out the message body.
- virtual std::vector< std::string > **getPropertyNames** () const =0 throw (CMSEException)
Retrieves the property names.
- virtual bool **propertyExists** (const std::string &name) const =0 throw (CMSEException)
Indicates whether or not a given property exists.
- virtual bool **getBooleanProperty** (const std::string &name) const =0 throw (MessageFormatException, CMSEException)
Gets a boolean property.
- virtual unsigned char **getByteProperty** (const std::string &name) const =0 throw (MessageFormatException, CMSEException)
Gets a byte property.
- virtual double **getDoubleProperty** (const std::string &name) const =0 throw (MessageFormatException, CMSEException)
Gets a double property.
- virtual float **getFloatProperty** (const std::string &name) const =0 throw (MessageFormatException, CMSEException)
Gets a float property.

- virtual int **getIntProperty** (const std::string &name) const =0 throw (MessageFormatException, CMSEException)
Gets a int property.
- virtual long long **getLongProperty** (const std::string &name) const =0 throw (MessageFormatException, CMSEException)
Gets a long property.
- virtual short **getShortProperty** (const std::string &name) const =0 throw (MessageFormatException, CMSEException)
Gets a short property.
- virtual std::string **getStringProperty** (const std::string &name) const =0 throw (MessageFormatException, CMSEException)
Gets a string property.
- virtual void **setBooleanProperty** (const std::string &name, bool value)=0 throw (MessageNotWriteableException, CMSEException)
Sets a boolean property.
- virtual void **setByteProperty** (const std::string &name, unsigned char value)=0 throw (MessageNotWriteableException, CMSEException)
Sets a byte property.
- virtual void **setDoubleProperty** (const std::string &name, double value)=0 throw (MessageNotWriteableException, CMSEException)
Sets a double property.
- virtual void **setFloatProperty** (const std::string &name, float value)=0 throw (MessageNotWriteableException, CMSEException)
Sets a float property.
- virtual void **setIntProperty** (const std::string &name, int value)=0 throw (MessageNotWriteableException, CMSEException)
Sets a int property.
- virtual void **setLongProperty** (const std::string &name, long long value)=0 throw (MessageNotWriteableException, CMSEException)
Sets a long property.
- virtual void **setShortProperty** (const std::string &name, short value)=0 throw (MessageNotWriteableException, CMSEException)
Sets a short property.
- virtual void **setStringProperty** (const std::string &name, const std::string &value)=0 throw (MessageNotWriteableException, CMSEException)
Sets a string property.
- virtual std::string **getCMSCorrelationID** () const =0 throw (CMSEException)
Gets the correlation ID for the message.

- virtual void **setCMSCorrelationID** (const std::string &correlationId)=0 throw (CMSEception)
Sets the correlation ID for the message.
- virtual int **getCMSDeliveryMode** () const =0 throw (CMSEception)
*Gets the **DeliveryMode** (p. 1386) for this message.*
- virtual void **setCMSDeliveryMode** (int mode)=0 throw (CMSEception)
*Sets the **DeliveryMode** (p. 1386) for this message.*
- virtual const **Destination** * **getCMSDestination** () const =0 throw (CMSEception)
*Gets the **Destination** (p. 1387) object for this message.*
- virtual void **setCMSDestination** (const **Destination** *destination)=0 throw (CMSEception)
*Sets the **Destination** (p. 1387) object for this message.*
- virtual long long **getCMSExpiration** () const =0 throw (CMSEception)
Gets the message's expiration value.
- virtual void **setCMSExpiration** (long long expireTime)=0 throw (CMSEception)
Sets the message's expiration value.
- virtual std::string **getCMSMessageID** () const =0 throw (CMSEception)
The CMSMessageID header field contains a value that uniquely identifies each message sent by a provider.
- virtual void **setCMSMessageID** (const std::string &id)=0 throw (CMSEception)
Sets the message ID.
- virtual int **getCMSPriority** () const =0 throw (CMSEception)
Gets the message priority level.
- virtual void **setCMSPriority** (int priority)=0 throw (CMSEception)
Sets the Priority Value for this message.
- virtual bool **getCMSRedelivered** () const =0 throw (CMSEception)
Gets an indication of whether this message is being redelivered.
- virtual void **setCMSRedelivered** (bool redelivered)=0 throw (CMSEception)
Specifies whether this message is being redelivered.
- virtual const **cms::Destination** * **getCMSReplyTo** () const =0 throw (CMSEception)
*Gets the **Destination** (p. 1387) object to which a reply to this message should be sent.*
- virtual void **setCMSReplyTo** (const **cms::Destination** *destination)=0 throw (CMSEception)
*Sets the **Destination** (p. 1387) object to which a reply to this message should be sent.*

- virtual long long **getCMSTimestamp** () const =0 throw (CMSEException)
Gets the message timestamp.
- virtual void **setCMSTimestamp** (long long timeStamp)=0 throw (CMSEException)
Sets the message timestamp.
- virtual std::string **getCMSType** () const =0 throw (CMSEException)
Gets the message type identifier supplied by the client when the message was sent.
- virtual void **setCMSType** (const std::string &type)=0 throw (CMSEException)
Sets the message type.

6.435.1 Detailed Description

Root of all messages. As in JMS, a message is comprised of 3 parts: CMS-specific headers, user-defined properties, and the body.

Message (p. 2036) Bodies

The CMS API defines four types of message bodies, each type is contained within its own **Message** (p. 2036) Interface definition.

- Stream - A **StreamMessage** (p. 2892) object's message body contains a stream of primitive values in the C++ language. It is filled and read sequentially. Unlike the **BytesMessage** (p. 874) type the values written to a **StreamMessage** (p. 2892) retain information on their type and rules for type conversion are enforced when reading back the values from the **Message** (p. 2036) Body.
- Map - A **MapMessage** (p. 1982) object's message body contains a set of name-value pairs, where names are std::string objects, and values are C++ primitives. The entries can be accessed sequentially or randomly by name. The **MapMessage** (p. 1982) makes no guarantee on the order of the elements within the **Message** (p. 2036) body.
- Text - A **TextMessage** (p. 2974) object's message body contains a std::string object. This message type can be used to transport plain-text messages, and XML messages.
- Bytes - A **BytesMessage** (p. 874) object's message body contains a stream of uninterpreted bytes. This message type is for literally encoding a body to match an existing message format. In many cases, it is possible to use one of the other body types, which are easier to use.

Message (p. 2036) Properties

Message (p. 2036) properties support the following conversion table. The marked cases must be supported. The unmarked cases must throw a **CMSEException** (p. 960). The String-to-primitive conversions may throw a runtime exception if the primitive's valueOf method does not accept the String as a valid representation of the primitive.

A value written as the row type can be read as the column type.

	boolean	byte	short	int	long	float	double	String
boolean	X							X
byte		X	X	X	X			X

short				X	X	X			X
int					X	X			X
long						X			X
float							X	X	X
double								X	X
String		X	X	X	X	X	X	X	X

When a **Message** (p. 2036) is delivered its properties are considered to be in a read-only mode and cannot be changed. Attempting to change the value of a delivered Message's properties will result in a **CMSEException** (p. 960) being thrown.

See also

JMS API

Since

1.0

6.435.2 Constructor & Destructor Documentation

6.435.2.1 `virtual cms::Message::~Message () [inline, virtual]`

6.435.3 Member Function Documentation

6.435.3.1 `virtual void cms::Message::acknowledge () const throw (
 IllegalStateException, CMSEException) [pure virtual]`

Acknowledges all consumed messages of the session of this consumed message.

All consumed CMS messages support the acknowledge method for use when a client has specified that its CMS session's consumed messages are to be explicitly acknowledged. By invoking acknowledge on a consumed message, a client acknowledges all messages consumed by the session that the message was delivered to.

Calls to acknowledge are ignored for both transacted sessions and sessions specified to use implicit acknowledgment modes.

A client may individually acknowledge each message as it is consumed, or it may choose to acknowledge messages as an application-defined group (which is done by calling acknowledge on the last received message of the group, thereby acknowledging all messages consumed by the session.)

Messages that have been received but not acknowledged may be redelivered.

Exceptions

CMSEException (p. 960) - if an internal error occurs.

IllegalStateException (p. 1621) - if this method is called on a closed session.

6.435.3.2 `virtual void cms::Message::clearBody () throw (CMSEException)
 [pure virtual]`

Clears out the body of the message.

This does not clear the headers or properties.

Exceptions

CMSEException (p. 960) - if an internal error occurs.

6.435.3.3 `virtual void cms::Message::clearProperties () throw (CMSEException)`
[pure virtual]

Clears out the message body.

Clearing a message's body does not clear its header values or property entries.

If this message body was read-only, calling this method leaves the message body in the same state as an empty body in a newly created message.

Exceptions

CMSEException (p. 960) - if an internal error occurs.

6.435.3.4 `virtual Message* cms::Message::clone () const` [pure virtual]

Clone this message exactly, returns a new instance that the caller is required to delete.

Returns

new copy of this message

Implemented in `cms::BytesMessage` (p. 877).

6.435.3.5 `virtual bool cms::Message::getBooleanProperty (const std::string & name) const throw (MessageFormatException, CMSEException)` [pure virtual]

Gets a boolean property.

Parameters

name The name of the property to retrieve.

Returns

The value for the named property.

Exceptions

CMSEException (p. 960) if the property does not exist.

MessageFormatException (p. 2141) - if this type conversion is invalid.

6.435.3.6 `virtual unsigned char cms::Message::getByteProperty (const std::string & name) const throw (MessageFormatException, CMSEException)`
[pure virtual]

Gets a byte property.

Parameters

name The name of the property to retrieve.

Returns

The value for the named property.

Exceptions

CMSException (p. 960) if the property does not exist.

MessageFormatException (p. 2141) - if this type conversion is invalid.

6.435.3.7 `virtual std::string cms::Message::getCMSCorrelationID () const throw (CMSException) [pure virtual]`

Gets the correlation ID for the message.

This method is used to return correlation ID values that are either provider-specific message IDs or application-specific String values.

Returns

string representation of the correlation Id

Exceptions

CMSException (p. 960) - if an internal error occurs.

6.435.3.8 `virtual int cms::Message::getCMSDeliveryMode () const throw (CMSException) [pure virtual]`

Gets the **DeliveryMode** (p. 1386) for this message.

Returns

DeliveryMode (p. 1386) enumerated value.

Exceptions

CMSException (p. 960) - if an internal error occurs.

6.435.3.9 `virtual const Destination* cms::Message::getCMSDestination () const throw (CMSException) [pure virtual]`

Gets the **Destination** (p. 1387) object for this message.

The CMSDestination header field contains the destination to which the message is being sent.

When a message is sent, this field is ignored. After completion of the send or publish method, the field holds the destination specified by the method.

When a message is received, its CMSDestination value must be equivalent to the value assigned when it was sent.

Returns

Destination (p. 1387) object

Exceptions

CMSException (p. 960) - if an internal error occurs.

6.435.3.10 `virtual long long cms::Message::getCMSExpiration () const throw (CMSException) [pure virtual]`

Gets the message's expiration value.

When a message is sent, the CMSExpiration header field is left unassigned. After completion of the send or publish method, it holds the expiration time of the message. This is the sum of the time-to-live value specified by the client and the GMT at the time of the send or publish.

If the time-to-live is specified as zero, CMSExpiration is set to zero to indicate that the message does not expire.

When a message's expiration time is reached, a provider should discard it. The CMS API does not define any form of notification of message expiration.

Clients should not receive messages that have expired; however, the CMS API does not guarantee that this will not happen.

Returns

the time the message expires, which is the sum of the time-to-live value specified by the client and the GMT at the time of the send

Exceptions

CMSException (p. 960) - if an internal error occurs.

6.435.3.11 `virtual std::string cms::Message::getCMSMessageID () const throw (CMSException) [pure virtual]`

The CMSMessageID header field contains a value that uniquely identifies each message sent by a provider.

When a message is sent, CMSMessageID can be ignored. When the send or publish method returns, it contains a provider-assigned value.

A CMSMessageID is a String value that should function as a unique key for identifying messages in a historical repository. The exact scope of uniqueness is provider-defined. It should at least cover all messages for a specific installation of a provider, where an installation is some connected set of message routers.

All CMSMessageID values must start with the prefix 'ID:'. Uniqueness of message ID values across different providers is not required.

Since message IDs take some effort to create and increase a message's size, some CMS providers may be able to optimize message overhead if they are given a hint that the message ID is not used by an application. By calling the **MessageProducer.setDisableMessageID** (p. 2195) method, a CMS client enables this potential optimization for all messages sent by that message producer. If the CMS provider accepts this hint, these messages must have the message ID set to null; if the provider ignores the hint, the message ID must be set to its normal unique value.

Returns

provider-assigned message id

Exceptions

CMSEException (p. 960) - if an internal error occurs.

6.435.3.12 `virtual int cms::Message::getCMSPriority () const throw (CMSEException) [pure virtual]`

Gets the message priority level.

The CMS API defines ten levels of priority value, with 0 as the lowest priority and 9 as the highest. In addition, clients should consider priorities 0-4 as gradations of normal priority and priorities 5-9 as gradations of expedited priority.

The CMS API does not require that a provider strictly implement priority ordering of messages; however, it should do its best to deliver expedited messages ahead of normal messages.

Returns

priority value

Exceptions

CMSEException (p. 960) - if an internal error occurs.

6.435.3.13 `virtual bool cms::Message::getCMSRedelivered () const throw (CMSEException) [pure virtual]`

Gets an indication of whether this message is being redelivered.

If a client receives a message with the CMSRedelivered field set, it is likely, but not guaranteed, that this message was delivered earlier but that its receipt was not acknowledged at that time.

Returns

true if this message is being redelivered

Exceptions

CMSEException (p. 960) - if an internal error occurs.

6.435.3.14 `virtual const cms::Destination* cms::Message::getCMSReplyTo () const throw (CMSEException) [pure virtual]`

Gets the **Destination** (p. 1387) object to which a reply to this message should be sent.

Returns

Destination (p. 1387) to which to send a response to this message

Exceptions

CMSEException (p. 960) - if an internal error occurs.

6.435.3.15 `virtual long long cms::Message::getCMSTimestamp () const throw (CMSEException) [pure virtual]`

Gets the message timestamp.

The CMSTimestamp header field contains the time a message was handed off to a provider to be sent. It is not the time the message was actually transmitted, because the actual send may occur later due to transactions or other client-side queuing of messages.

When a message is sent, CMSTimestamp is ignored. When the send or publish method returns, it contains a time value somewhere in the interval between the call and the return. The value is in the format of a normal millis time value in the Java programming language.

Since timestamps take some effort to create and increase a message's size, some CMS providers may be able to optimize message overhead if they are given a hint that the timestamp is not used by an application. By calling the MessageProducer.setDisableMessageTimestamp method, a CMS client enables this potential optimization for all messages sent by that message producer. If the CMS provider accepts this hint, these messages must have the timestamp set to zero; if the provider ignores the hint, the timestamp must be set to its normal value.

Returns

the message timestamp

Exceptions

CMSEException (p. 960) - if an internal error occurs.

6.435.3.16 `virtual std::string cms::Message::getCMSType () const throw (CMSEException) [pure virtual]`

Gets the message type identifier supplied by the client when the message was sent.

Returns

the message type

See also

setCMSType (p. 2052)

Exceptions

CMSEException (p. 960) - if an internal error occurs.

6.435.3.17 `virtual double cms::Message::getDoubleProperty (const std::string & name) const throw (MessageFormatException, CMSEException) [pure virtual]`

Gets a double property.

Parameters

name The name of the property to retrieve.

Returns

The value for the named property.

Exceptions

CMSEException (p. 960) if the property does not exist.

MessageFormatException (p. 2141) - if this type conversion is invalid.

6.435.3.18 `virtual float cms::Message::getFloatProperty (const std::string & name) const throw (MessageFormatException, CMSEException)`
[pure virtual]

Gets a float property.

Parameters

name The name of the property to retrieve.

Returns

The value for the named property.

Exceptions

CMSEException (p. 960) if the property does not exist.

MessageFormatException (p. 2141) - if this type conversion is invalid.

6.435.3.19 `virtual int cms::Message::getIntProperty (const std::string & name) const throw (MessageFormatException, CMSEException)`
[pure virtual]

Gets a int property.

Parameters

name The name of the property to retrieve.

Returns

The value for the named property.

Exceptions

CMSEException (p. 960) if the property does not exist.

MessageFormatException (p. 2141) - if this type conversion is invalid.

6.435.3.20 `virtual long long cms::Message::getLongProperty (const std::string & name) const throw (MessageFormatException, CMSEException)`
[pure virtual]

Gets a long property.

Parameters

name The name of the property to retrieve.

Returns

The value for the named property.

Exceptions

CMSEException (p. 960) if the property does not exist.

MessageFormatException (p. 2141) - if this type conversion is invalid.

6.435.3.21 `virtual std::vector<std::string> cms::Message::getPropertyNames ()
const throw (CMSEException) [pure virtual]`

Retrieves the property names.

Returns

The complete set of property names currently in this message.

Exceptions

CMSEException (p. 960) - if an internal error occurs.

6.435.3.22 `virtual short cms::Message::getShortProperty (const std::string &
name) const throw (MessageFormatException, CMSEException)
[pure virtual]`

Gets a short property.

Parameters

name The name of the property to retrieve.

Returns

The value for the named property.

Exceptions

CMSEException (p. 960) if the property does not exist.

MessageFormatException (p. 2141) - if this type conversion is invalid.

6.435.3.23 `virtual std::string cms::Message::getStringProperty (const std::string
& name) const throw (MessageFormatException, CMSEException)
[pure virtual]`

Gets a string property.

Parameters

name The name of the property to retrieve.

Returns

The value for the named property.

Exceptions

CMSEException (p. 960) if the property does not exist.

MessageFormatException (p. 2141) - if this type conversion is invalid.

6.435.3.24 `virtual bool cms::Message::propertyExists (const std::string & name) const throw (CMSEException)` [pure virtual]

Indicates whether or not a given property exists.

Parameters

name The name of the property to look up.

Returns

True if the property exists in this message.

Exceptions

CMSEException (p. 960) - if an internal error occurs.

6.435.3.25 `virtual void cms::Message::setBooleanProperty (const std::string & name, bool value) throw (MessageNotWriteableException, CMSEException)` [pure virtual]

Sets a boolean property.

Parameters

name The name of the property to retrieve.

value The value for the named property.

Exceptions

CMSEException (p. 960) - if the name is an empty string.

MessageNotWriteableException (p. 2188) - if properties are read-only

6.435.3.26 `virtual void cms::Message::setByteProperty (const std::string & name, unsigned char value) throw (MessageNotWriteableException, CMSEException)` [pure virtual]

Sets a byte property.

Parameters

name The name of the property to retrieve.

value The value for the named property.

Exceptions

CMSException (p. 960) - if the name is an empty string.

MessageNotWriteableException (p. 2188) - if properties are read-only

6.435.3.27 `virtual void cms::Message::setCMSCorrelationID (const std::string & correlationId) throw (CMSException) [pure virtual]`

Sets the correlation ID for the message.

A client can use the CMSCorrelationID header field to link one message with another. A typical use is to link a response message with its request message.

CMSCorrelationID can hold one of the following:

- A provider-specific message ID
- An application-specific String
- A provider-native byte[] value

Since each message sent by a CMS provider is assigned a message ID value, it is convenient to link messages via message ID. All message ID values must start with the 'ID:' prefix.

In some cases, an application (made up of several clients) needs to use an application-specific value for linking messages. For instance, an application may use CMSCorrelationID to hold a value referencing some external information. Application-specified values must not start with the 'ID:' prefix; this is reserved for provider-generated message ID values.

If a provider supports the native concept of correlation ID, a CMS client may need to assign specific CMSCorrelationID values to match those expected by clients that do not use the CMS API. A byte[] value is used for this purpose. CMS providers without native correlation ID values are not required to support byte[] values. The use of a byte[] value for CMSCorrelationID is non-portable.

Parameters

correlationId The message ID of a message being referred to.

Exceptions

CMSException (p. 960) - if an internal error occurs.

6.435.3.28 `virtual void cms::Message::setCMSDeliveryMode (int mode) throw (CMSException) [pure virtual]`

Sets the **DeliveryMode** (p.1386) for this message.

CMS providers set this field when a message is sent. This method can be used to change the value for a message that has been received.

Parameters

mode **DeliveryMode** (p.1386) enumerated value.

Exceptions

CMSEException (p. 960) - if an internal error occurs.

6.435.3.29 `virtual void cms::Message::setCMSDestination (const Destination *
destination) throw (CMSEException) [pure virtual]`

Sets the **Destination** (p.1387) object for this message.

CMS providers set this field when a message is sent. This method can be used to change the value for a message that has been received.

Parameters

destination **Destination** (p.1387) Object

Exceptions

CMSEException (p. 960) - if an internal error occurs.

6.435.3.30 `virtual void cms::Message::setCMSExpiration (long long expireTime
) throw (CMSEException) [pure virtual]`

Sets the message's expiration value.

CMS providers set this field when a message is sent. This method can be used to change the value for a message that has been received.

Parameters

expireTime the message's expiration time

Exceptions

CMSEException (p. 960) - if an internal error occurs.

6.435.3.31 `virtual void cms::Message::setCMSMessageID (const std::string & id
) throw (CMSEException) [pure virtual]`

Sets the message ID.

CMS providers set this field when a message is sent. This method can be used to change the value for a message that has been received.

Parameters

id the ID of the message

Exceptions

CMSEException (p. 960) - if an internal error occurs.

6.435.3.32 `virtual void cms::Message::setCMSPriority (int priority) throw (CMSEException) [pure virtual]`

Sets the Priority Value for this message.

CMS providers set this field when a message is sent. This method can be used to change the value for a message that has been received.

Parameters

priority priority value for this message

Exceptions

CMSEException (p. 960) - if an internal error occurs.

6.435.3.33 `virtual void cms::Message::setCMSRedelivered (bool redelivered) throw (CMSEException) [pure virtual]`

Specifies whether this message is being redelivered.

This field is set at the time the message is delivered. This method can be used to change the value for a message that has been received.

Parameters

redelivered boolean redelivered value

Exceptions

CMSEException (p. 960) - if an internal error occurs.

6.435.3.34 `virtual void cms::Message::setCMSReplyTo (const cms::Destination * destination) throw (CMSEException) [pure virtual]`

Sets the **Destination** (p. 1387) object to which a reply to this message should be sent.

The CMSReplyTo header field contains the destination where a reply to the current message should be sent. If it is null, no reply is expected. The destination may be either a **Queue** (p. 2510) object or a **Topic** (p. 3014) object.

Messages sent with a null CMSReplyTo value may be a notification of some event, or they may just be some data the sender thinks is of interest.

Messages with a CMSReplyTo value typically expect a response. A response is optional; it is up to the client to decide. These messages are called requests. A message sent in response to a request is called a reply.

In some cases a client may wish to match a request it sent earlier with a reply it has just received. The client can use the CMSCorrelationID header field for this purpose.

Parameters

destination **Destination** (p. 1387) to which to send a response to this message

Exceptions

CMSEException (p. 960) - if an internal error occurs.

6.435.3.35 `virtual void cms::Message::setCMSTimestamp (long long timeStamp) throw (CMSException)` [pure virtual]

Sets the message timestamp.

CMS providers set this field when a message is sent. This method can be used to change the value for a message that has been received.

Parameters

timeStamp integer time stamp value

Exceptions

CMSException (p. 960) - if an internal error occurs.

6.435.3.36 `virtual void cms::Message::setCMSType (const std::string & type) throw (CMSException)` [pure virtual]

Sets the message type.

Some CMS providers use a message repository that contains the definitions of messages sent by applications. The CMSType header field may reference a message's definition in the provider's repository.

The CMS API does not define a standard message definition repository, nor does it define a naming policy for the definitions it contains.

Some messaging systems require that a message type definition for each application message be created and that each message specify its type. In order to work with such CMS providers, CMS clients should assign a value to CMSType, whether the application makes use of it or not. This ensures that the field is properly set for those providers that require it.

To ensure portability, CMS clients should use symbolic values for CMSType that can be configured at installation time to the values defined in the current provider's message repository. If string literals are used, they may not be valid type names for some CMS providers.

Parameters

type the message type

See also

`getCMSType` (p. 2045)

Exceptions

CMSException (p. 960) - if an internal error occurs.

6.435.3.37 `virtual void cms::Message::setDoubleProperty (const std::string & name, double value) throw (MessageNotWriteableException, CMSException)` [pure virtual]

Sets a double property.

Parameters

name The name of the property to retrieve.

value The value for the named property.

Exceptions

CMSEException (p. 960) - if the name is an empty string.

MessageNotWriteableException (p. 2188) - if properties are read-only

6.435.3.38 `virtual void cms::Message::setFloatProperty (const std::string & name, float value) throw (MessageNotWriteableException, CMSEException) [pure virtual]`

Sets a float property.

Parameters

name The name of the property to retrieve.

value The value for the named property.

Exceptions

CMSEException (p. 960) - if the name is an empty string.

MessageNotWriteableException (p. 2188) - if properties are read-only

6.435.3.39 `virtual void cms::Message::setIntProperty (const std::string & name, int value) throw (MessageNotWriteableException, CMSEException) [pure virtual]`

Sets a int property.

Parameters

name The name of the property to retrieve.

value The value for the named property.

Exceptions

CMSEException (p. 960) - if the name is an empty string.

MessageNotWriteableException (p. 2188) - if properties are read-only

6.435.3.40 `virtual void cms::Message::setLongProperty (const std::string & name, long long value) throw (MessageNotWriteableException, CMSEException) [pure virtual]`

Sets a long property.

Parameters

name The name of the property to retrieve.

value The value for the named property.

Exceptions

CMSEException (p. 960) - if the name is an empty string.

MessageNotWriteableException (p. 2188) - if properties are read-only

6.435.3.41 `virtual void cms::Message::setShortProperty (const std::string & name, short value) throw (MessageNotWriteableException, CMSEException) [pure virtual]`

Sets a short property.

Parameters

name The name of the property to retrieve.

value The value for the named property.

Exceptions

CMSEException (p. 960) - if the name is an empty string.

MessageNotWriteableException (p. 2188) - if properties are read-only

6.435.3.42 `virtual void cms::Message::setStringProperty (const std::string & name, const std::string & value) throw (MessageNotWriteableException, CMSEException) [pure virtual]`

Sets a string property.

Parameters

name The name of the property to retrieve.

value The value for the named property.

Exceptions

CMSEException (p. 960) - if the name is an empty string.

MessageNotWriteableException (p. 2188) - if properties are read-only

The documentation for this class was generated from the following file:

- `src/main/cms/Message.h`

6.436 activemq::commands::MessageAck Class Reference

```
#include <src/main/activemq/commands/MessageAck.h>
```

Inheritance diagram for `activemq::commands::MessageAck`:

Public Member Functions

- **MessageAck** ()
- virtual **~MessageAck** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **MessageAck * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1372) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **ActiveMQDestination** > & **getDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &destination)
- virtual const **Pointer**< **TransactionId** > & **getTransactionId** () const
- virtual **Pointer**< **TransactionId** > & **getTransactionId** ()
- virtual void **setTransactionId** (const **Pointer**< **TransactionId** > &transactionId)
- virtual const **Pointer**< **ConsumerId** > & **getConsumerId** () const
- virtual **Pointer**< **ConsumerId** > & **getConsumerId** ()
- virtual void **setConsumerId** (const **Pointer**< **ConsumerId** > &consumerId)
- virtual unsigned char **getAckType** () const
- virtual void **setAckType** (unsigned char ackType)
- virtual const **Pointer**< **MessageId** > & **getFirstMessageId** () const
- virtual **Pointer**< **MessageId** > & **getFirstMessageId** ()
- virtual void **setFirstMessageId** (const **Pointer**< **MessageId** > &firstMessageId)
- virtual const **Pointer**< **MessageId** > & **getLastMessageId** () const
- virtual **Pointer**< **MessageId** > & **getLastMessageId** ()
- virtual void **setLastMessageId** (const **Pointer**< **MessageId** > &lastMessageId)
- virtual int **getMessageCount** () const
- virtual void **setMessageCount** (int messageCount)
- virtual bool **isMessageAck** () const
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor *visitor)
throw (exceptions::ActiveMQException)
Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_MESSAGEACK** = 22

Protected Member Functions

- **MessageAck** (const MessageAck &)
- **MessageAck & operator=** (const MessageAck &)

Protected Attributes

- **Pointer< ActiveMQDestination > destination**
- **Pointer< TransactionId > transactionId**
- **Pointer< ConsumerId > consumerId**
- **unsigned char ackType**
- **Pointer< MessageId > firstMessageId**
- **Pointer< MessageId > lastMessageId**
- **int messageCount**

6.436.1 Constructor & Destructor Documentation

6.436.1.1 **activemq::commands::MessageAck::MessageAck** (const MessageAck &) [inline, protected]

6.436.1.2 **activemq::commands::MessageAck::MessageAck** ()

6.436.1.3 **virtual activemq::commands::MessageAck::~~MessageAck** () [virtual]

6.436.2 Member Function Documentation

6.436.2.1 **virtual MessageAck* activemq::commands::MessageAck::cloneDataStructure** () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1373).

6.436.2.2 **virtual void activemq::commands::MessageAck::copyDataStructure** (const DataStructure * *src*) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 598).

6.436.2.3 `virtual bool activemq::commands::MessageAck::equals (const
DataStructure * value) const [virtual]`

Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 598).

6.436.2.4 `virtual unsigned char activemq::commands::MessageAck::getAckType () const [virtual]`

6.436.2.5 `virtual const Pointer<ConsumerId>& activemq::commands::MessageAck::getConsumerId () const [virtual]`

6.436.2.6 `virtual Pointer<ConsumerId>& activemq::commands::MessageAck::getConsumerId () [virtual]`

6.436.2.7 `virtual unsigned char activemq::commands::MessageAck::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataStructure** (p. 1372) type copy.

Implements **activemq::commands::DataStructure** (p. 1375).

- 6.436.2.8** virtual Pointer<ActiveMQDestination>& activemq::commands::MessageAck::getDestination ()
[virtual]
- 6.436.2.9** virtual const Pointer<ActiveMQDestination>& activemq::commands::MessageAck::getDestination () const [virtual]
- 6.436.2.10** virtual const Pointer<MessageId>& activemq::commands::MessageAck::getFirstMessageId () const [virtual]
- 6.436.2.11** virtual Pointer<MessageId>& activemq::commands::MessageAck::getFirstMessageId () [virtual]
- 6.436.2.12** virtual const Pointer<MessageId>& activemq::commands::MessageAck::getLastMessageId () const [virtual]
- 6.436.2.13** virtual Pointer<MessageId>& activemq::commands::MessageAck::getLastMessageId () [virtual]
- 6.436.2.14** virtual int activemq::commands::MessageAck::getMessageCount () const [virtual]
- 6.436.2.15** virtual const Pointer<TransactionId>& activemq::commands::MessageAck::getTransactionId () const [virtual]
- 6.436.2.16** virtual Pointer<TransactionId>& activemq::commands::MessageAck::getTransactionId () [virtual]
- 6.436.2.17** virtual bool activemq::commands::MessageAck::isMessageAck () const [inline, virtual]

Returns

an answer of true to the `isMessageAck()` (p. 2058) query.

Reimplemented from `activemq::commands::BaseCommand` (p. 600).

- 6.436.2.18 `MessageAck& activemq::commands::MessageAck::operator= (const MessageAck &) [inline, protected]`
- 6.436.2.19 `virtual void activemq::commands::MessageAck::setAckType (unsigned char ackType) [virtual]`
- 6.436.2.20 `virtual void activemq::commands::MessageAck::setConsumerId (const Pointer< ConsumerId > & consumerId) [virtual]`
- 6.436.2.21 `virtual void activemq::commands::MessageAck::setDestination (const Pointer< ActiveMQDestination > & destination) [virtual]`
- 6.436.2.22 `virtual void activemq::commands::MessageAck::setFirstMessageId (const Pointer< MessageId > & firstMessageId) [virtual]`
- 6.436.2.23 `virtual void activemq::commands::MessageAck::setLastMessageId (const Pointer< MessageId > & lastMessageId) [virtual]`
- 6.436.2.24 `virtual void activemq::commands::MessageAck::setMessageCount (int messageCount) [virtual]`
- 6.436.2.25 `virtual void activemq::commands::MessageAck::setTransactionId (const Pointer< TransactionId > & transactionId) [virtual]`
- 6.436.2.26 `virtual std::string activemq::commands::MessageAck::toString () const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1372) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 602).

- 6.436.2.27 `virtual Pointer<Command> activemq::commands::MessageAck::visit (activemq::state::CommandVisitor * visitor) throw (exceptions::ActiveMQException) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

Returns

a **Response** (p. 2611) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 995).

6.436.3 Field Documentation

- 6.436.3.1 unsigned char activemq::commands::MessageAck::ackType [protected]
- 6.436.3.2 Pointer<ConsumerId> activemq::commands::MessageAck::consumerId [protected]
- 6.436.3.3 Pointer<ActiveMQDestination> activemq::commands::MessageAck::destination [protected]
- 6.436.3.4 Pointer<MessageId> activemq::commands::MessageAck::firstMessageId [protected]
- 6.436.3.5 const unsigned char activemq::commands::MessageAck::ID_ - MESSAGEACK = 22 [static]
- 6.436.3.6 Pointer<MessageId> activemq::commands::MessageAck::lastMessageId [protected]
- 6.436.3.7 int activemq::commands::MessageAck::messageCount [protected]
- 6.436.3.8 Pointer<TransactionId> activemq::commands::MessageAck::transactionId [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/MessageAck.h

6.437 activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller Class Reference

Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2060).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/MessageAckMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller:

Public Member Functions

- **MessageAckMarshaller** ()
- virtual **~MessageAckMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.437.1 Detailed Description

Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2060). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.437.2 Constructor & Destructor Documentation

6.437.2.1 activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller::MessageAckMarshaller () [inline]

6.437.2.2 virtual activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller::~~MessageAckMarshaller () [inline, virtual]

6.437.3 Member Function Documentation

6.437.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1331).

6.437.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller::getDataSetType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1336).

6.437.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataSet * dataSet, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 631).

6.437.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataSet * dataSet, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 632).

6.437.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 633).

6.437.3.6 `virtual void activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 634).

6.437.3.7 `virtual void activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 635).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/MessageAckMarshaller.h`

6.438 `activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller` Class Reference

Marshaling code for Open Wire Format for `MessageAckMarshaller` (p. 2064).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/MessageAckMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller`:

Public Member Functions

- `MessageAckMarshaller ()`
- `virtual ~MessageAckMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaler.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)`

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.438.1 Detailed Description

Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2064). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.438.2 Constructor & Destructor Documentation

6.438.2.1 **activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller::MessageAckMarshaller** () [inline]

6.438.2.2 **virtual**
activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller::~~MessageAckMarshaller () [inline, virtual]

6.438.3 Member Function Documentation

6.438.3.1 **virtual commands::DataStructure*** **activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller::createObject** () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1331).

6.438.3.2 **virtual unsigned char** **activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller::getDataStructureType** () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1336).

6.438.3.3 virtual void activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 624).

6.438.3.4 virtual void activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 625).

6.438.3.5 virtual int activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 626).

```
6.438.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::MessageAckMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 628).

```
6.438.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::MessageAckMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 629).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/MessageAckMarshaller.h`

6.439 `activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller` Class Reference

Marshaling code for Open Wire Format for `MessageAckMarshaller` (p. 2068).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/MessageAckMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller`:

Public Member Functions

- `MessageAckMarshaller ()`
- `virtual ~MessageAckMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaller.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write a object instance to data output stream.
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.439.1 Detailed Description

Marshaling code for Open Wire Format for **MessageAckMarshaller** (p.2068). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.439.2 Constructor & Destructor Documentation

6.439.2.1 activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller::MessageAckMarshaller () [inline]

6.439.2.2 virtual
activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller::~~MessageAckMarshaller () [inline, virtual]

6.439.3 Member Function Documentation

6.439.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1331).

6.439.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1336).

6.439.3.3 virtual void activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 604).

6.439.3.4 virtual void activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 605).

6.439.3.5 virtual int activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 606).

```
6.439.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::MessageAckMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 608).

```
6.439.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::MessageAckMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 609).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/MessageAckMarshaller.h`

6.440 activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller Class Reference

Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2072).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/MessageAckMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller**:

Public Member Functions

- **MessageAckMarshaller** ()
- virtual **~MessageAckMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.440.1 Detailed Description

Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2072). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.440.2 Constructor & Destructor Documentation

6.440.2.1 activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller::MessageAckMarshaller () [inline]

6.440.2.2 virtual
activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller::~~MessageAckMarshaller () [inline, virtual]

6.440.3 Member Function Documentation

6.440.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1331).

6.440.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1336).

6.440.3.3 virtual void activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 618).

6.440.3.4 virtual void activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 619).

6.440.3.5 virtual int activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 620).

```
6.440.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::MessageAckMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 621).

```
6.440.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::MessageAckMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 622).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/MessageAckMarshaller.h`

6.441 activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller Class Reference

Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2076).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/MessageAckMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller**:

Public Member Functions

- **MessageAckMarshaller** ()
- virtual **~MessageAckMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.441.1 Detailed Description

Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2076). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.441.2 Constructor & Destructor Documentation

6.441.2.1 activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller::MessageAckMarshaller () [inline]

6.441.2.2 virtual
activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller::~~MessageAckMarshaller () [inline, virtual]

6.441.3 Member Function Documentation

6.441.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1331).

6.441.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1336).

6.441.3.3 virtual void activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 611).

6.441.3.4 virtual void activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 612).

6.441.3.5 virtual int activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 613).

```
6.441.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::MessageAckMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 614).

```
6.441.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::MessageAckMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 615).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**MessageAckMarshaller.h**

6.442 cms::MessageConsumer Class Reference

A client uses a **MessageConsumer** (p. 2080) to received messages from a destination.

#include <src/main/cms/MessageConsumer.h>

Inheritance diagram for cms::MessageConsumer:

Public Member Functions

- virtual **~MessageConsumer** ()
- virtual **Message * receive** ()=0 throw (**CMSEException**)
*Synchronously Receive a **Message** (p. 2036).*
- virtual **Message * receive** (int millisecs)=0 throw (**CMSEException**)
*Synchronously Receive a **Message** (p. 2036), time out after defined interval.*
- virtual **Message * receiveNoWait** ()=0 throw (**CMSEException**)
*Receive a **Message** (p. 2036), does not wait if there isn't a new message to read, returns NULL if nothing read.*
- virtual void **setMessageListener** (**MessageListener** *listener)=0 throw (**CMSEException**)
*Sets the **MessageListener** (p. 2165) that this class will send notifs on.*
- virtual **MessageListener * getMessageListener** () const =0 throw (**CMSEException**)
*Gets the **MessageListener** (p. 2165) that this class will send new **Message** (p. 2036) notification events to.*
- virtual std::string **getMessageSelector** () const =0 throw (cms::CMSEException)
Gets this message consumer's message selector expression.

6.442.1 Detailed Description

A client uses a **MessageConsumer** (p. 2080) to received messages from a destination. A client may either synchronously receive a message consumer's messages or have the consumer asynchronously deliver them as they arrive.

For synchronous receipt, a client can request the next message from a message consumer using one of its **receive** methods. There are several variations of **receive** that allow a client to poll or wait for the next message.

For asynchronous delivery, a client can register a **MessageListener** (p. 2165) object with a message consumer. As messages arrive at the message consumer, it delivers them by calling the **MessageListener** (p. 2165)'s **onMessage** method.

When the **MessageConsumer**'s **close** method is called the method can block while an asynchronous message delivery is in progress or until a **receive** operation completes. A blocked consumer in a **receive** call will return a Null when the **close** method is called.

See also

MessageListener (p. 2165)

Since

1.0

6.442.2 Constructor & Destructor Documentation

6.442.2.1 **virtual cms::MessageConsumer::~MessageConsumer ()** [inline, virtual]

6.442.3 Member Function Documentation

6.442.3.1 **virtual MessageListener* cms::MessageConsumer::getMessageListener ()** **const throw (CMSEException)** [pure virtual]

Gets the **MessageListener** (p. 2165) that this class will send new **Message** (p. 2036) notification events to.

Returns

The listener of messages received by this consumer

Exceptions

CMSEException (p. 960) - If an internal error occurs.

Implemented in **activemq::cmsutil::CachedConsumer** (p. 889).

6.442.3.2 **virtual std::string cms::MessageConsumer::getMessageSelector ()** **const throw (cms::CMSEException)** [pure virtual]

Gets this message consumer's message selector expression.

Returns

This Consumer's selector expression or "".

Exceptions

CMSEException (p. 960) - If an internal error occurs.

Implemented in **activemq::cmsutil::CachedConsumer** (p. 889).

6.442.3.3 `virtual Message* cms::MessageConsumer::receive (int millisecs)
throw (CMSEException) [pure virtual]`

Synchronously Receive a **Message** (p. 2036), time out after defined interval.

Returns null if nothing read.

Returns

new message which the caller owns and must delete.

Exceptions

CMSEException (p. 960) - If an internal error occurs.

Implemented in **activemq::cmsutil::CachedConsumer** (p. 890).

6.442.3.4 `virtual Message* cms::MessageConsumer::receive () throw (CMSEException) [pure virtual]`

Synchronously Receive a **Message** (p. 2036).

Returns

new message which the caller owns and must delete.

Exceptions

CMSEException (p. 960) - If an internal error occurs.

Implemented in **activemq::cmsutil::CachedConsumer** (p. 889).

6.442.3.5 `virtual Message* cms::MessageConsumer::receiveNoWait () throw (CMSEException) [pure virtual]`

Receive a **Message** (p. 2036), does not wait if there isn't a new message to read, returns NULL if nothing read.

Returns

new message which the caller owns and must delete.

Exceptions

CMSEException (p. 960) - If an internal error occurs.

Implemented in **activemq::cmsutil::CachedConsumer** (p. 890).

6.442.3.6 `virtual void cms::MessageConsumer::setMessageListener (MessageListener * listener) throw (CMSEException) [pure virtual]`

Sets the **MessageListener** (p. 2165) that this class will send notifs on.

Parameters

listener The listener of messages received by this consumer.

Exceptions

CMSEException (p. 960) - If an internal error occurs.

Implemented in **activemq::cmsutil::CachedConsumer** (p. 890).

The documentation for this class was generated from the following file:

- `src/main/cms/MessageConsumer.h`

6.443 activemq::cmsutil::MessageCreator Class Reference

Creates the user-defined message to be sent by the `CmsTemplate` (p. 969).

```
#include <src/main/activemq/cmsutil/MessageCreator.h>
```

Public Member Functions

- `virtual ~MessageCreator ()`
- `virtual cms::Message * createMessage (cms::Session *session)=0 throw (cms::CMSEException)`
Creates a message from the given session.

6.443.1 Detailed Description

Creates the user-defined message to be sent by the `CmsTemplate` (p. 969).

6.443.2 Constructor & Destructor Documentation

- 6.443.2.1** `virtual activemq::cmsutil::MessageCreator::~MessageCreator ()`
`[inline, virtual]`

6.443.3 Member Function Documentation

- 6.443.3.1** `virtual cms::Message* activemq::cmsutil::MessageCreator::createMessage (cms::Session * session) throw (cms::CMSEException) [pure virtual]`

Creates a message from the given session.

Parameters

session the CMS Session

Exceptions

cms::CMSEException (p. 960) if thrown by CMS API methods

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/MessageCreator.h

6.444 activemq::commands::MessageDispatch Class Reference

```
#include <src/main/activemq/commands/MessageDispatch.h>
```

Inheritance diagram for activemq::commands::MessageDispatch:

Public Member Functions

- **MessageDispatch** ()
- virtual **~MessageDispatch** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **MessageDispatch * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1372) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **ConsumerId** > & **getConsumerId** () const
- virtual **Pointer**< **ConsumerId** > **getConsumerId** ()
- virtual void **setConsumerId** (const **Pointer**< **ConsumerId** > &consumerId)
- virtual const **Pointer**< **ActiveMQDestination** > & **getDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &destination)
- virtual const **Pointer**< **Message** > & **getMessage** () const
- virtual **Pointer**< **Message** > & **getMessage** ()
- virtual void **setMessage** (const **Pointer**< **Message** > &message)
- virtual int **getRedeliveryCounter** () const
- virtual void **setRedeliveryCounter** (int redeliveryCounter)
- virtual bool **isMessageDispatch** () const
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor *visitor)
throw (exceptions::ActiveMQException)
Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_MESSAGE_DISPATCH** = 21

Protected Member Functions

- **MessageDispatch** (const **MessageDispatch** &)
- **MessageDispatch** & **operator=** (const **MessageDispatch** &)

Protected Attributes

- **Pointer**< **ConsumerId** > **consumerId**
- **Pointer**< **ActiveMQDestination** > **destination**
- **Pointer**< **Message** > **message**
- int **redeliveryCounter**

6.444.1 Constructor & Destructor Documentation

6.444.1.1 **activemq::commands::MessageDispatch::MessageDispatch** (const **MessageDispatch** &) [inline, protected]

6.444.1.2 **activemq::commands::MessageDispatch::MessageDispatch** ()

6.444.1.3 **virtual activemq::commands::MessageDispatch::~~MessageDispatch** () [virtual]

6.444.2 Member Function Documentation

6.444.2.1 **virtual MessageDispatch* activemq::commands::MessageDispatch::cloneDataStructure** () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1373).

6.444.2.2 **virtual void activemq::commands::MessageDispatch::copyDataStructure** (const **DataStructure** * *src*) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 598).

6.444.2.3 `virtual bool activemq::commands::MessageDispatch::equals (const DataStructure * value) const` [virtual]

Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 598).

6.444.2.4 `virtual Pointer<ConsumerId>& activemq::commands::MessageDispatch::getConsumerId ()` [virtual]

6.444.2.5 `virtual const Pointer<ConsumerId>& activemq::commands::MessageDispatch::getConsumerId () const` [virtual]

6.444.2.6 `virtual unsigned char activemq::commands::MessageDispatch::getDataStructureType () const` [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataStructure** (p. 1372) type copy.

Implements **activemq::commands::DataStructure** (p. 1375).

- 6.444.2.7 `virtual const Pointer<ActiveMQDestination>& activemq::commands::MessageDispatch::getDestination () const` [virtual]
- 6.444.2.8 `virtual Pointer<ActiveMQDestination>& activemq::commands::MessageDispatch::getDestination ()` [virtual]
- 6.444.2.9 `virtual const Pointer<Message>& activemq::commands::MessageDispatch::getMessage () const` [virtual]
- 6.444.2.10 `virtual Pointer<Message>& activemq::commands::MessageDispatch::getMessage ()` [virtual]
- 6.444.2.11 `virtual int activemq::commands::MessageDispatch::getRedeliveryCounter () const` [virtual]
- 6.444.2.12 `virtual bool activemq::commands::MessageDispatch::isMessageDispatch () const` [inline, virtual]

Returns

an answer of true to the `isMessageDispatch()` (p. 2087) query.

Reimplemented from `activemq::commands::BaseCommand` (p. 600).

- 6.444.2.13 `MessageDispatch& activemq::commands::MessageDispatch::operator= (const MessageDispatch &)` [inline, protected]
- 6.444.2.14 `virtual void activemq::commands::MessageDispatch::setConsumerId (const Pointer< ConsumerId > & consumerId)` [virtual]
- 6.444.2.15 `virtual void activemq::commands::MessageDispatch::setDestination (const Pointer< ActiveMQDestination > & destination)` [virtual]
- 6.444.2.16 `virtual void activemq::commands::MessageDispatch::setMessage (const Pointer< Message > & message)` [virtual]
- 6.444.2.17 `virtual void activemq::commands::MessageDispatch::setRedeliveryCounter (int redeliveryCounter)` [virtual]
- 6.444.2.18 `virtual std::string activemq::commands::MessageDispatch::toString () const` [virtual]

Returns a string containing the information for this **DataStructure** (p. 1372) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from `activemq::commands::BaseCommand` (p. 602).

```
6.444.2.19 virtual Pointer<Command> ac-
            tivemq::commands::MessageDispatch::visit (
            activemq::state::CommandVisitor * visitor ) throw (
            exceptions::ActiveMQException ) [virtual]
```

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

Returns

a **Response** (p. 2611) to the visitor being called or NULL if no response.

Implements `activemq::commands::Command` (p. 995).

6.444.3 Field Documentation

- 6.444.3.1 `Pointer<ConsumerId> activemq::commands::MessageDispatch::consumerId`
[protected]
- 6.444.3.2 `Pointer<ActiveMQDestination> activemq::commands::MessageDispatch::destination`
[protected]
- 6.444.3.3 `const unsigned char activemq::commands::MessageDispatch::ID_ -
MESSAGEDISPATCH = 21` [static]
- 6.444.3.4 `Pointer<Message> activemq::commands::MessageDispatch::message`
[protected]
- 6.444.3.5 `int activemq::commands::MessageDispatch::redeliveryCounter`
[protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/MessageDispatch.h`

6.445 activemq::core::MessageDispatchChannel Class Reference

```
#include <src/main/activemq/core/MessageDispatchChannel.h>
```

Inheritance diagram for `activemq::core::MessageDispatchChannel`:

Public Member Functions

- **MessageDispatchChannel** ()
- virtual **~MessageDispatchChannel** ()
- void **enqueue** (const **Pointer**< **MessageDispatch** > &message)
Add a Message to the Channel behind all pending message.
- void **enqueueFirst** (const **Pointer**< **MessageDispatch** > &message)
Add a message to the front of the Channel.
- bool **isEmpty** () const
- bool **isClosed** () const
- bool **isRunning** () const
- **Pointer**< **MessageDispatch** > **dequeue** (long long timeout) throw (exceptions::ActiveMQException)
Used to get an enqueued message.
- **Pointer**< **MessageDispatch** > **dequeueNoWait** ()
Used to get an enqueued message if there is one queued right now.
- **Pointer**< **MessageDispatch** > **peek** () const
Peek in the Queue and return the first message in the Channel without removing it from the channel.
- void **start** ()
Starts dispatch of messages from the Channel.
- void **stop** ()
Stops dispatch of message from the Channel.
- void **close** ()
Close this channel no messages will be dispatched after this method is called.
- void **clear** ()
Clear the Channel, all pending messages are removed.
- int **size** () const
- std::vector< **Pointer**< **MessageDispatch** > > **removeAll** ()
Remove all messages that are currently in the Channel and return them as a list of Messages.
- virtual void **lock** () throw (decaf::lang::exceptions::RuntimeException)
Locks the object.
- virtual bool **tryLock** () throw (decaf::lang::exceptions::RuntimeException)
Attempts to Lock the object, if the lock is already held by another thread than this method returns false.
- virtual void **unlock** () throw (decaf::lang::exceptions::RuntimeException)
Unlocks the object.

- virtual void **wait** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (long long millisecs) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (long long millisecs, int nanos) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **notify** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)
Signals a waiter on this object that it can now wake up and continue.
- virtual void **notifyAll** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)
Signals the waiters on this object that it can now wake up and continue.

6.445.1 Constructor & Destructor Documentation

6.445.1.1 **activemq::core::MessageDispatchChannel::MessageDispatchChannel ()**

6.445.1.2 **virtual
activemq::core::MessageDispatchChannel::~~MessageDispatchChannel ()** [virtual]

6.445.2 Member Function Documentation

6.445.2.1 **void activemq::core::MessageDispatchChannel::clear ()**

Clear the Channel, all pending messages are removed.

6.445.2.2 **void activemq::core::MessageDispatchChannel::close ()**

Close this channel no messages will be dispatched after this method is called.

6.445.2.3 **Pointer<MessageDispatch> activemq::core::MessageDispatchChannel::dequeue (long
long *timeout*) throw (exceptions::ActiveMQException)**

Used to get an enqueued message.

The amount of time this method blocks is based on the timeout value. - if timeout== -1 then it blocks until a message is received. - if timeout==0 then it tries to not block at all, it returns

a message if it is available - if timeout>0 then it blocks up to timeout amount of time. Expired messages will be consumed by this method.

Returns

null if we timeout or if the consumer is closed.

Exceptions

ActiveMQException

6.445.2.4 `Pointer<MessageDispatch> activemq::core::MessageDispatchChannel::dequeueNoWait ()`

Used to get an enqueued message if there is one queued right now.

If there is no waiting message then this method returns Null.

Returns

a message if there is one in the queue.

6.445.2.5 `void activemq::core::MessageDispatchChannel::enqueue (const Pointer< MessageDispatch > & message)`

Add a Message to the Channel behind all pending message.

Parameters

message - The message to add to the Channel.

6.445.2.6 `void activemq::core::MessageDispatchChannel::enqueueFirst (const Pointer< MessageDispatch > & message)`

Add a message to the front of the Channel.

Parameters

message - The Message to add to the front of the Channel.

6.445.2.7 `bool activemq::core::MessageDispatchChannel::isClosed () const [inline]`

Returns

has the Queue been closed.

6.445.2.8 `bool activemq::core::MessageDispatchChannel::isEmpty () const`

Returns

true if there are no messages in the Channel.

6.445.2.9 `bool activemq::core::MessageDispatchChannel::isRunning () const`
 `[inline]`

Returns

true if the Channel currently running and will dequeue message.

6.445.2.10 `virtual void activemq::core::MessageDispatchChannel::lock () throw (`
 `decaf::lang::exceptions::RuntimeException) [inline, virtual]`

Locks the object.

Exceptions

RuntimeException if an error occurs while locking the object.

Implements `decaf::util::concurrent::Synchronizable` (p. 2932).

6.445.2.11 `virtual void activemq::core::MessageDispatchChannel::notify`
 `() throw (decaf::lang::exceptions::RuntimeException,`
 `decaf::lang::exceptions::IllegalMonitorStateException) [inline,`
 `virtual]`

Signals a waiter on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

RuntimeException if an error occurs while notifying one of the waiting threads.

Implements `decaf::util::concurrent::Synchronizable` (p. 2933).

6.445.2.12 `virtual void activemq::core::MessageDispatchChannel::notifyAll`
 `() throw (decaf::lang::exceptions::RuntimeException,`
 `decaf::lang::exceptions::IllegalMonitorStateException) [inline,`
 `virtual]`

Signals the waiters on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

RuntimeException if an error occurs while notifying the waiting threads.

Implements `decaf::util::concurrent::Synchronizable` (p. 2934).

6.445.2.13 `Pointer<MessageDispatch> activemq::core::MessageDispatchChannel::peek () const`

Peek in the Queue and return the first message in the Channel without removing it from the channel.

Returns

a message if there is one in the queue.

6.445.2.14 `std::vector< Pointer<MessageDispatch> > activemq::core::MessageDispatchChannel::removeAll ()`

Remove all messages that are currently in the Channel and return them as a list of Messages.

Returns

a list of Messages that was previously in the Channel.

6.445.2.15 `int activemq::core::MessageDispatchChannel::size () const`

Returns

the number of Messages currently in the Channel.

6.445.2.16 `void activemq::core::MessageDispatchChannel::start ()`

Starts dispatch of messages from the Channel.

6.445.2.17 `void activemq::core::MessageDispatchChannel::stop ()`

Stops dispatch of message from the Channel.

6.445.2.18 `virtual bool activemq::core::MessageDispatchChannel::tryLock () throw (decaf::lang::exceptions::RuntimeException) [inline, virtual]`

Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

Returns

true if the lock was acquired, false if it is already held by another thread.

Exceptions

RuntimeException if an error occurs while locking the object.

Implements `decaf::util::concurrent::Synchronizable` (p. 2935).

6.445.2.19 `virtual void activemq::core::MessageDispatchChannel::unlock ()
throw (decaf::lang::exceptions::RuntimeException) [inline, virtual]`

Unlocks the object.

Exceptions

RuntimeException if an error occurs while unlocking the object.

Implements `decaf::util::concurrent::Synchronizable` (p. 2936).

6.445.2.20 `virtual void activemq::core::MessageDispatchChannel::wait (long
long millisecs) throw (decaf::lang::exceptions::RuntimeException,
decaf::lang::exceptions::IllegalMonitorStateException,
decaf::lang::exceptions::InterruptedException) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval.

Parameters

millisecs the time in milliseconds to wait, or WAIT_INFINITE

Exceptions

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements `decaf::util::concurrent::Synchronizable` (p. 2938).

6.445.2.21 `virtual void activemq::core::MessageDispatchChannel::wait
(long long millisecs, int nanos) throw
(decaf::lang::exceptions::RuntimeException,
decaf::lang::exceptions::IllegalArgumentException,
decaf::lang::exceptions::IllegalMonitorStateException,
decaf::lang::exceptions::InterruptedException) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

Parameters

millisecs the time in milliseconds to wait, or WAIT_INFINITE

nanos additional time in nanoseconds with a range of 0-999999

Exceptions

IllegalArgumentException if an error occurs or the nanos argument is not in the range of [0-999999]

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2940).

```
6.445.2.22  virtual void activemq::core::MessageDispatchChannel::wait
            (      ) throw ( decaf::lang::exceptions::RuntimeException,
            decaf::lang::exceptions::IllegalMonitorStateException,
            decaf::lang::exceptions::InterruptedException ) [inline, virtual]
```

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling.

Exceptions

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2937).

The documentation for this class was generated from the following file:

- src/main/activemq/core/MessageDispatchChannel.h

6.446 activemq::wireformat::openwire::marshal::v2::MessageDispatchMa

Class Reference

Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2095).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/MessageDispatchMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller**:

Public Member Functions

- **MessageDispatchMarshaller** ()
- virtual **~MessageDispatchMarshaller** ()
- virtual **commands::DataStructure * createObject** () const

Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.446.1 Detailed Description

Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p.2095). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.446.2 Constructor & Destructor Documentation

6.446.2.1 `activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller::MessageDispatchMarshaller () [inline]`

6.446.2.2 `virtual activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller::~~MessageDispatchMarshaller () [inline, virtual]`

6.446.3 Member Function Documentation

6.446.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.446.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.446.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 631).

6.446.3.4 virtual void activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 632).

6.446.3.5 virtual int activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 633).

6.446.3.6 virtual void activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 634).

```
6.446.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 635).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/MessageDispatchMarshaller.h`

6.447 `activemq::wireformat::openwire::marshal::v4::MessageDispatchMa` Class Reference

Marshaling code for Open Wire Format for `MessageDispatchMarshaller` (p. 2099).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/MessageDispatchMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller`:

Public Member Functions

- **MessageDispatchMarshaller** ()
- virtual **~MessageDispatchMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.447.1 Detailed Description

Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p.2099). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.447.2 Constructor & Destructor Documentation

6.447.2.1 `activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller::MessageDispatch`
() [inline]

6.447.2.2 `virtual`
`activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller::~~MessageDispatch`
() [inline, virtual]

6.447.3 Member Function Documentation

6.447.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller::createObject`
() const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.447.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller::getDataStructure`
() const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.447.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller::looseMarshal`
(`OpenWireFormat * wireFormat`, `commands::DataStructure * dataStructure`, `decaf::io::DataOutputStream * dataOut`) throw (`decaf::io::IOException`) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 618).

6.447.3.4 virtual void activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 619).

6.447.3.5 virtual int activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 620).

6.447.3.6 virtual void activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 621).

```
6.447.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 622).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/MessageDispatchMarshaller.h`

6.448 `activemq::wireformat::openwire::marshal::v3::MessageDispatchMa` Class Reference

Marshaling code for Open Wire Format for `MessageDispatchMarshaller` (p. 2103).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/MessageDispatchMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller`:

Public Member Functions

- **MessageDispatchMarshaller** ()
- virtual **~MessageDispatchMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.448.1 Detailed Description

Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2103). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.448.2 Constructor & Destructor Documentation

6.448.2.1 `activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller::MessageDispatchMarshaller () [inline]`

6.448.2.2 `virtual activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller::~~MessageDispatchMarshaller () [inline, virtual]`

6.448.3 Member Function Documentation

6.448.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.448.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.448.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 604).

```

6.448.3.4 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]

```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 605).

```

6.448.3.5 virtual int ac-
tivemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, utils::BooleanStream * bs ) throw (
decaf::io::IOException ) [virtual]

```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 606).

```

6.448.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 608).

```
6.448.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 609).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/MessageDispatchMarshaller.h`

6.449 `activemq::wireformat::openwire::marshal::v1::MessageDispatchMa` Class Reference

Marshaling code for Open Wire Format for `MessageDispatchMarshaller` (p. 2107).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/MessageDispatchMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller`:

Public Member Functions

- **MessageDispatchMarshaller** ()
- virtual **~MessageDispatchMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.449.1 Detailed Description

Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2107). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.449.2 Constructor & Destructor Documentation

6.449.2.1 `activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller::MessageDispatch`
() [inline]

6.449.2.2 `virtual`
`activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller::~~MessageDispatch`
() [inline, virtual]

6.449.3 Member Function Documentation

6.449.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller::createObject`
() const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.449.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller::getDataStructure`
() const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.449.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller::looseMarshal`
(`OpenWireFormat * wireFormat`, `commands::DataStructure * dataStructure`, `decaf::io::DataOutputStream * dataOut`) throw (`decaf::io::IOException`) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 611).


```

6.449.3.4 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]

```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 612).

```

6.449.3.5 virtual int ac-
tivemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, utils::BooleanStream * bs ) throw (
decaf::io::IOException ) [virtual]

```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 613).

```

6.449.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 614).

```
6.449.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 615).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/MessageDispatchMarshaller.h`

6.450 `activemq::wireformat::openwire::marshal::v5::MessageDispatchMa` Class Reference

Marshaling code for Open Wire Format for `MessageDispatchMarshaller` (p. 2111).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/MessageDispatchMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller`:

Public Member Functions

- **MessageDispatchMarshaller** ()
- virtual **~MessageDispatchMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.450.1 Detailed Description

Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p.2111). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.450.2 Constructor & Destructor Documentation

6.450.2.1 `activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller::MessageDispatchMarshaller () [inline]`

6.450.2.2 `virtual activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller::~~MessageDispatchMarshaller () [inline, virtual]`

6.450.3 Member Function Documentation

6.450.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.450.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.450.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 624).

6.450.3.4 virtual void activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 625).

6.450.3.5 virtual int activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 626).

6.450.3.6 virtual void activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions

- IOException* if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 628).

```
6.450.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 629).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/MessageDispatchMarshaller.h`

6.451 `activemq::commands::MessageDispatchNotification` Class Reference

```
#include <src/main/activemq/commands/MessageDispatchNotification.h>
```

Inheritance diagram for `activemq::commands::MessageDispatchNotification`:

Public Member Functions

- **MessageDispatchNotification** ()
- virtual **~MessageDispatchNotification** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **MessageDispatchNotification * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1372) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **ConsumerId** > & **getConsumerId** () const
- virtual **Pointer**< **ConsumerId** > & **getConsumerId** ()
- virtual void **setConsumerId** (const **Pointer**< **ConsumerId** > &consumerId)
- virtual const **Pointer**< **ActiveMQDestination** > & **getDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &destination)
- virtual long long **getDeliverySequenceId** () const
- virtual void **setDeliverySequenceId** (long long deliverySequenceId)
- virtual const **Pointer**< **MessageId** > & **getMessageId** () const
- virtual **Pointer**< **MessageId** > & **getMessageId** ()
- virtual void **setMessageId** (const **Pointer**< **MessageId** > &messageId)
- virtual bool **isMessageDispatchNotification** () const
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor *visitor)
throw (exceptions::ActiveMQException)
Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID _MESSAGEDISPATCHNOTIFICATION** = 90

Protected Member Functions

- **MessageDispatchNotification** (const **MessageDispatchNotification** &)
- **MessageDispatchNotification** & **operator=** (const **MessageDispatchNotification** &)

Protected Attributes

- `Pointer< ConsumerId > consumerId`
- `Pointer< ActiveMQDestination > destination`
- `long long deliverySequenceId`
- `Pointer< MessageId > messageId`

6.451.1 Constructor & Destructor Documentation

6.451.1.1 `activemq::commands::MessageDispatchNotification::MessageDispatchNotification (const MessageDispatchNotification &) [inline, protected]`

6.451.1.2 `activemq::commands::MessageDispatchNotification::MessageDispatchNotification ()`

6.451.1.3 `virtual
activemq::commands::MessageDispatchNotification::~~MessageDispatchNotification () [virtual]`

6.451.2 Member Function Documentation

6.451.2.1 `virtual MessageDispatchNotification* activemq::commands::MessageDispatchNotification::cloneDataStructure () const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1373).

6.451.2.2 `virtual void activemq::commands::MessageDispatchNotification::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

Reimplemented from `activemq::commands::BaseCommand` (p. 598).

6.451.2.3 `virtual bool activemq::commands::MessageDispatchNotification::equals (const DataStructure * value) const [virtual]`

Compares the `DataStructure` (p. 1372) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 598).

6.451.2.4 virtual **Pointer<ConsumerId>&** **activemq::commands::MessageDispatchNotification::getConsumerId** ()
[virtual]

6.451.2.5 virtual const **Pointer<ConsumerId>&** **activemq::commands::MessageDispatchNotification::getConsumerId** ()
const [virtual]

6.451.2.6 virtual unsigned char **activemq::commands::MessageDispatchNotification::getDataStructureType**
() const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataStructure** (p. 1372) type copy.

Implements **activemq::commands::DataStructure** (p. 1375).

6.451.2.7 virtual long long **activemq::commands::MessageDispatchNotification::getDeliverySequenceId**
() const [virtual]

6.451.2.8 virtual const **Pointer<ActiveMQDestination>&** **activemq::commands::MessageDispatchNotification::getDestination** ()
const [virtual]

6.451.2.9 virtual **Pointer<ActiveMQDestination>&** **activemq::commands::MessageDispatchNotification::getDestination** ()
[virtual]

6.451.2.10 virtual **Pointer<MessageId>&** **activemq::commands::MessageDispatchNotification::getMessageId** ()
[virtual]

6.451.2.11 virtual const **Pointer<MessageId>&** **activemq::commands::MessageDispatchNotification::getMessageId** ()
const [virtual]

6.451.2.12 virtual bool **activemq::commands::MessageDispatchNotification::isMessageDispatchNotification**
() const [inline, virtual]

Returns

an answer of true to the **isMessageDispatchNotification()** (p. 2118) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 601).

- 6.451.2.13 `MessageDispatchNotification& activemq::commands::MessageDispatchNotification::operator= (const MessageDispatchNotification &)` [inline, protected]
- 6.451.2.14 `virtual void activemq::commands::MessageDispatchNotification::setConsumerId (const Pointer< ConsumerId > & consumerId)` [virtual]
- 6.451.2.15 `virtual void activemq::commands::MessageDispatchNotification::setDeliverySequenceId (long long deliverySequenceId)` [virtual]
- 6.451.2.16 `virtual void activemq::commands::MessageDispatchNotification::setDestination (const Pointer< ActiveMQDestination > & destination)` [virtual]
- 6.451.2.17 `virtual void activemq::commands::MessageDispatchNotification::setMessageId (const Pointer< MessageId > & messageId)` [virtual]
- 6.451.2.18 `virtual std::string activemq::commands::MessageDispatchNotification::toString () const` [virtual]

Returns a string containing the information for this **DataStructure** (p.1372) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from `activemq::commands::BaseCommand` (p.602).

- 6.451.2.19 `virtual Pointer<Command> activemq::commands::MessageDispatchNotification::visit (activemq::state::CommandVisitor * visitor) throw (exceptions::ActiveMQException)` [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

Returns

a **Response** (p.2611) to the visitor being called or NULL if no response.

Implements `activemq::commands::Command` (p.995).

- 6.451.3.1 `Pointer<ConsumerId>` `activemq::commands::MessageDispatchNotification::consumerId`
[protected]
- 6.451.3.2 `long long` `activemq::commands::MessageDispatchNotification::deliverySequenceId`
[protected]
- 6.451.3.3 `Pointer<ActiveMQDestination>` `activemq::commands::MessageDispatchNotification::destination`
[protected]
- 6.451.3.4 `const unsigned char` `activemq::commands::MessageDispatchNotification::ID` -
`MESSAGEDISPATCHNOTIFICATION = 90` [static]
- 6.451.3.5 `Pointer<MessageId>` `activemq::commands::MessageDispatchNotification::messageId`
[protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/MessageDispatchNotification.h`

6.452 activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller Class Reference

Marshaling code for Open Wire Format for `MessageDispatchNotificationMarshaller` (p. 2120).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/MessageDispatchNotificationMarshaller
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller`:

Public Member Functions

- `MessageDispatchNotificationMarshaller ()`
- `virtual ~MessageDispatchNotificationMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaler.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.452.1 Detailed Description

Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p.2120). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.452.2 Constructor & Destructor Documentation

6.452.2.1 `activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller::MessageDispatchNotificationMarshaller () [inline]`

6.452.2.2 `virtual activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller::~MessageDispatchNotificationMarshaller () [inline, virtual]`

6.452.3 Member Function Documentation

6.452.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller::create () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1331).

6.452.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller::getDataStructureType() const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1336).

6.452.3.3 virtual void activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller::marshal(const OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 631).

6.452.3.4 virtual void activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller::unmarshal(const OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 632).

6.452.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller::tightM`
`(OpenWireFormat * wireFormat, commands::DataStructure`
`* dataStructure, utils::BooleanStream * bs) throw (`
`decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 633).

6.452.3.6 `virtual void ac-`
`tivemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller::tightM`
`(OpenWireFormat * wireFormat, commands::DataStructure`
`* dataStructure, decaf::io::DataOutputStream * dataOut,`
`utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 634).

6.452.3.7 `virtual void ac-`
`tivemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller::tightU`
`(OpenWireFormat * wireFormat, commands::DataStructure`
`* dataStructure, decaf::io::DataInputStream * dataIn,`
`utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 635).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/MessageDispatchNotificationMarshaller.h`

6.453 activemq::wireformat::openwire::marshal::v4::MessageDispatchNo Class Reference

Marshaling code for Open Wire Format for `MessageDispatchNotificationMarshaller` (p. 2124).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/MessageDispatchNotificationMarshaller
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller`:

Public Member Functions

- `MessageDispatchNotificationMarshaller ()`
- `virtual ~MessageDispatchNotificationMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaler.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.453.1 Detailed Description

Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p.2124). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.453.2 Constructor & Destructor Documentation

6.453.2.1 **activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller::Me**
() [inline]

6.453.2.2 **virtual**
activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller::~~M
() [inline, virtual]

6.453.3 Member Function Documentation

6.453.3.1 **virtual commands::DataStructure* ac-**
tivemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller::create
() const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1331).

6.453.3.2 **virtual unsigned char ac-**
tivemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller::getDa
() const [virtual]

Get the Data Structure Type that identifies this Marshaler.

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.453.3.3 `virtual void ac-`
`tivemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller::loose`
`(OpenWireFormat * wireFormat, commands::DataStructure *`
`dataStructure, decaf::io::DataOutputStream * dataOut) throw (`
`decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller`
 (p. 618).

6.453.3.4 `virtual void ac-`
`tivemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller::loose`
`(OpenWireFormat * wireFormat, commands::DataStructure *`
`dataStructure, decaf::io::DataInputStream * dataIn) throw (`
`decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller`
 (p. 619).

6.453.3.5 `virtual int ac-`
`tivemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller::tight`
`(OpenWireFormat * wireFormat, commands::DataStructure`
`* dataStructure, utils::BooleanStream * bs) throw (`
`decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 620).

```
6.453.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller::tightU
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 621).

```
6.453.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller::tightU
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 622).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/MessageDispatchNotificationMarshaller.h`

6.454 activemq::wireformat::openwire::marshal::v3::MessageDispatchNo Class Reference

Marshaling code for Open Wire Format for `MessageDispatchNotificationMarshaller` (p. 2128).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/MessageDispatchNotificationMarshaller
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller`:

Public Member Functions

- `MessageDispatchNotificationMarshaller ()`
- `virtual ~MessageDispatchNotificationMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaller.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write a object instance to data output stream.

- virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.454.1 Detailed Description

Marshaling code for Open Wire Format for MessageDispatchNotificationMarshaller (p.2128). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.454.2 Constructor & Destructor Documentation

6.454.2.1 activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller::MessageDispatchNotificationMarshaller () [inline]

6.454.2.2 virtual activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller::~MessageDispatchNotificationMarshaller () [inline, virtual]

6.454.3 Member Function Documentation

6.454.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller::createDataStructure () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p.1331).

6.454.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p.1336).

6.454.3.3 **virtual void ac-**
 tivemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller::looseM
 (OpenWireFormat * *wireFormat*, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * *dataOut*) throw (
 decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller**
 (p. 604).

6.454.3.4 **virtual void ac-**
 tivemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller::looseU
 (OpenWireFormat * *wireFormat*, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * *dataIn*) throw (
 decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller**
 (p. 605).

6.454.3.5 **virtual int ac-**
 tivemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller::tightM
 (OpenWireFormat * *wireFormat*, commands::DataStructure
 * *dataStructure*, utils::BooleanStream * *bs*) throw (
 decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 606).

```
6.454.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller::tightM
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 608).

```
6.454.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller::tightU
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 609).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/MessageDispatchNotificationMarshaller.h`

6.455 **activemq::wireformat::openwire::marshal::v1::MessageDispatchNo** **Class Reference**

Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 2132).

`#include <src/main/activemq/wireformat/openwire/marshal/v1/MessageDispatchNotificationMarshaller`

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller**:

Public Member Functions

- **MessageDispatchNotificationMarshaller** ()
- virtual **~MessageDispatchNotificationMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshall an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.455.1 Detailed Description

Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p.2132). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.455.2 Constructor & Destructor Documentation

6.455.2.1 activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller::MessageDispatchNotificationMarshaller () [inline]

6.455.2.2 virtual activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller::~MessageDispatchNotificationMarshaller () [inline, virtual]

6.455.3 Member Function Documentation

6.455.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller::createDataStructure () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1331).

6.455.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1336).

6.455.3.3 `virtual void ac-`
 tivemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller::looseM
 (OpenWireFormat * *wireFormat*, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * *dataOut*) throw (
 decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller`
 (p. 611).

6.455.3.4 `virtual void ac-`
 tivemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller::looseU
 (OpenWireFormat * *wireFormat*, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * *dataIn*) throw (
 decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller`
 (p. 612).

6.455.3.5 `virtual int ac-`
 tivemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller::tightM
 (OpenWireFormat * *wireFormat*, commands::DataStructure
 * *dataStructure*, utils::BooleanStream * *bs*) throw (
 decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 613).

```
6.455.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller::tightM
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 614).

```
6.455.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller::tightU
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 615).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/MessageDispatchNotificationMarshaller.h`

6.456 **activemq::wireformat::openwire::marshal::v5::MessageDispatchNo** **Class Reference**

Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 2136).

`#include <src/main/activemq/wireformat/openwire/marshal/v5/MessageDispatchNotificationMarshaller`

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller**:

Public Member Functions

- **MessageDispatchNotificationMarshaller** ()
- virtual **~MessageDispatchNotificationMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshall an object instance from the data input stream.

- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)`

Write a object instance to data output stream.

6.456.1 Detailed Description

Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p.2136). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.456.2 Constructor & Destructor Documentation

6.456.2.1 `activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller::MessageDispatchNotificationMarshaller () [inline]`

6.456.2.2 `virtual activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller::~MessageDispatchNotificationMarshaller () [inline, virtual]`

6.456.3 Member Function Documentation

6.456.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller::createDataStructure () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1331).

6.456.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1336).

6.456.3.3 virtual void ac-
 tivemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller::looseM
 (OpenWireFormat * *wireFormat*, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * *dataOut*) throw (
 decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller`
 (p. 624).

6.456.3.4 virtual void ac-
 tivemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller::looseU
 (OpenWireFormat * *wireFormat*, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * *dataIn*) throw (
 decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller`
 (p. 625).

6.456.3.5 virtual int ac-
 tivemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller::tightM
 (OpenWireFormat * *wireFormat*, commands::DataStructure
 * *dataStructure*, utils::BooleanStream * *bs*) throw (
 decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 626).

```
6.456.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller::tightM
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 628).

```
6.456.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller::tightU
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 629).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v5/MessageDispatchNotificationMarshaller.h

6.457 cms::MessageEOFException Class Reference

This exception must be thrown when an unexpected end of stream has been reached when a **StreamMessage** (p. 2892) or **BytesMessage** (p. 874) is being read.

```
#include <src/main/cms/MessageEOFException.h>
```

Inheritance diagram for cms::MessageEOFException:

Public Member Functions

- **MessageEOFException** () throw ()
- **MessageEOFException** (const **MessageEOFException** &ex) throw ()
- **MessageEOFException** (const std::string &message, const std::exception *cause) throw ()
- **MessageEOFException** (const std::string &message, const std::exception *cause, const std::vector< std::pair< std::string, int > > &stackTrace) throw ()
- virtual ~**MessageEOFException** () throw ()

6.457.1 Detailed Description

This exception must be thrown when an unexpected end of stream has been reached when a **StreamMessage** (p. 2892) or **BytesMessage** (p. 874) is being read.

Since

1.3

6.457.2 Constructor & Destructor Documentation

- 6.457.2.1 `cms::MessageEOFException::MessageEOFException () throw ()`
- 6.457.2.2 `cms::MessageEOFException::MessageEOFException (const MessageEOFException & ex) throw ()`
- 6.457.2.3 `cms::MessageEOFException::MessageEOFException (const std::string & message, const std::exception * cause) throw ()`
- 6.457.2.4 `cms::MessageEOFException::MessageEOFException (const std::string & message, const std::exception * cause, const std::vector< std::pair< std::string, int > > & stackTrace) throw ()`
- 6.457.2.5 `virtual cms::MessageEOFException::~MessageEOFException () throw () [virtual]`

The documentation for this class was generated from the following file:

- `src/main/cms/MessageEOFException.h`

6.458 cms::MessageFormatException Class Reference

This exception must be thrown when a CMS client attempts to use a data type not supported by a message or attempts to read data in a message as the wrong type.

```
#include <src/main/cms/MessageFormatException.h>
```

Inheritance diagram for cms::MessageFormatException:

Public Member Functions

- `MessageFormatException () throw ()`
- `MessageFormatException (const MessageFormatException &ex) throw ()`
- `MessageFormatException (const std::string &message, const std::exception *cause) throw ()`
- `MessageFormatException (const std::string &message, const std::exception *cause, const std::vector< std::pair< std::string, int > > &stackTrace) throw ()`
- `virtual ~MessageFormatException () throw ()`

6.458.1 Detailed Description

This exception must be thrown when a CMS client attempts to use a data type not supported by a message or attempts to read data in a message as the wrong type. It must also be thrown when equivalent type errors are made with message property values. For example, this exception must be thrown if `StreamMessage.readShort` (p. 2899) is used to read a boolean value.

Since

1.3

6.458.2 Constructor & Destructor Documentation

- 6.458.2.1 `cms::MessageFormatException::MessageFormatException () throw ()`
- 6.458.2.2 `cms::MessageFormatException::MessageFormatException (const MessageFormatException & ex) throw ()`
- 6.458.2.3 `cms::MessageFormatException::MessageFormatException (const std::string & message, const std::exception * cause) throw ()`
- 6.458.2.4 `cms::MessageFormatException::MessageFormatException (const std::string & message, const std::exception * cause, const std::vector< std::pair< std::string, int > > & stackTrace) throw ()`
- 6.458.2.5 `virtual cms::MessageFormatException::~MessageFormatException () throw () [virtual]`

The documentation for this class was generated from the following file:

- `src/main/cms/MessageFormatException.h`

6.459 activemq::commands::MessageId Class Reference

```
#include <src/main/activemq/commands/MessageId.h>
```

Inheritance diagram for `activemq::commands::MessageId`:

Public Types

- `typedef decaf::lang::PointerComparator< MessageId > COMPARATOR`

Public Member Functions

- `MessageId ()`
- `MessageId (const MessageId &other)`
- `virtual ~MessageId ()`
- `virtual unsigned char getDataStructureType () const`
Get the unique identifier that this object and its own Marshaler share.
- `virtual MessageId * cloneDataStructure () const`
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- `virtual void copyDataStructure (const DataStructure *src)`
Copy the contents of the passed object into this object's members, overwriting any existing data.
- `virtual std::string toString () const`
*Returns a string containing the information for this **DataStructure** (p. 1372) such as its type and value of its elements.*

- virtual bool **equals** (const **DataSet** *value) const
*Compares the **DataSet** (p. 1372) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **ProducerId** > & **getProducerId** () const
- virtual **Pointer**< **ProducerId** > & **getProducerId** ()
- virtual void **setProducerId** (const **Pointer**< **ProducerId** > &producerId)
- virtual long long **getProducerSequenceId** () const
- virtual void **setProducerSequenceId** (long long producerSequenceId)
- virtual long long **getBrokerSequenceId** () const
- virtual void **setBrokerSequenceId** (long long brokerSequenceId)
- virtual int **compareTo** (const **MessageId** &value) const
- virtual bool **equals** (const **MessageId** &value) const
- virtual bool **operator==** (const **MessageId** &value) const
- virtual bool **operator<** (const **MessageId** &value) const
- **MessageId** & **operator=** (const **MessageId** &other)

Static Public Attributes

- static const unsigned char **ID_MESSAGEID** = 110

Protected Attributes

- **Pointer**< **ProducerId** > producerId
- long long producerSequenceId
- long long brokerSequenceId

6.459.1 Member Typedef Documentation

- 6.459.1.1 **typedef** decaf::lang::PointerComparator<MessageId>
activemq::commands::MessageId::COMPARATOR

6.459.2 Constructor & Destructor Documentation

- 6.459.2.1 **activemq::commands::MessageId::MessageId** ()
- 6.459.2.2 **activemq::commands::MessageId::MessageId** (const **MessageId** & *other*)
- 6.459.2.3 **virtual** **activemq::commands::MessageId::~~MessageId** () [virtual]

6.459.3 Member Function Documentation

- 6.459.3.1 **virtual** **MessageId*** **activemq::commands::MessageId::cloneDataSet**
() const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

6.459.3.2 `virtual int activemq::commands::MessageId::compareTo (const MessageId & value) const` [virtual]

6.459.3.3 `virtual void activemq::commands::MessageId::copyDataStructure (const DataStructure * src)` [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

6.459.3.4 `virtual bool activemq::commands::MessageId::equals (const DataStructure * value) const` [virtual]

Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

6.459.3.5 `virtual bool activemq::commands::MessageId::equals (const MessageId & value) const` [virtual]

6.459.3.6 `virtual long long activemq::commands::MessageId::getBrokerSequenceId () const` [virtual]

6.459.3.7 `virtual unsigned char activemq::commands::MessageId::getDataStructureType () const` [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataStructure** (p. 1372) type copy.

- 6.459.3.8 `virtual const Pointer<ProducerId>& activemq::commands::MessageId::getProducerId () const`
[virtual]
- 6.459.3.9 `virtual Pointer<ProducerId>& activemq::commands::MessageId::getProducerId ()`
[virtual]
- 6.459.3.10 `virtual long long activemq::commands::MessageId::getProducerSequenceId () const` [virtual]
- 6.459.3.11 `virtual bool activemq::commands::MessageId::operator< (const MessageId & value) const` [virtual]
- 6.459.3.12 `MessageId& activemq::commands::MessageId::operator= (const MessageId & other)`
- 6.459.3.13 `virtual bool activemq::commands::MessageId::operator== (const MessageId & value) const` [virtual]
- 6.459.3.14 `virtual void activemq::commands::MessageId::setBrokerSequenceId (long long brokerSequenceId)` [virtual]
- 6.459.3.15 `virtual void activemq::commands::MessageId::setProducerId (const Pointer< ProducerId > & producerId)` [virtual]
- 6.459.3.16 `virtual void activemq::commands::MessageId::setProducerSequenceId (long long producerSequenceId)` [virtual]
- 6.459.3.17 `virtual std::string activemq::commands::MessageId::toString () const`
[virtual]

Returns a string containing the information for this **DataStructure** (p.1372) such as its type and value of its elements.

Returns

formatted string useful for debugging.

6.459.4 Field Documentation

- 6.459.4.1 `long long activemq::commands::MessageId::brokerSequenceId`
[protected]
- 6.459.4.2 `const unsigned char activemq::commands::MessageId::ID_MESSAGEID`
`= 110` [static]
- 6.459.4.3 `Pointer<ProducerId> activemq::commands::MessageId::producerId`
[protected]
- 6.459.4.4 `long long activemq::commands::MessageId::producerSequenceId`
[protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/MessageId.h`

6.460 activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller Class Reference

Marshaling code for Open Wire Format for `MessageIdMarshaller` (p. 2146).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/MessageIdMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller`:

Public Member Functions

- `MessageIdMarshaller ()`
- `virtual ~MessageIdMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaler.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.460.1 Detailed Description

Marshaling code for Open Wire Format for **MessageIdMarshaller** (p.2146). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.460.2 Constructor & Destructor Documentation

6.460.2.1 **activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller::MessageIdMarshaller** () [inline]

6.460.2.2 **virtual**
activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller::~~MessageIdMarshaller () [inline, virtual]

6.460.3 Member Function Documentation

6.460.3.1 **virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller::createObject** () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1331).

6.460.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller::getDataStructureType** () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.460.3.3 virtual void `activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller::looseMarshal` (`OpenWireFormat * wireFormat`, `commands::DataStructure * dataStructure`, `decaf::io::DataOutputStream * dataOut`) throw (`decaf::io::IOException`) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1342).

6.460.3.4 virtual void `activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller::looseUnmarshal` (`OpenWireFormat * wireFormat`, `commands::DataStructure * dataStructure`, `decaf::io::DataInputStream * dataIn`) throw (`decaf::io::IOException`) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1348).

6.460.3.5 virtual int `activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller::tightMarshal1` (`OpenWireFormat * wireFormat`, `commands::DataStructure * dataStructure`, `utils::BooleanStream * bs`) throw (`decaf::io::IOException`) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1354).

```
6.460.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::MessageIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1360).

```
6.460.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::MessageIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1366).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/MessageIdMarshaller.h`

6.461 activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller Class Reference

Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2150).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/MessageIdMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller**:

Public Member Functions

- **MessageIdMarshaller** ()
- virtual **~MessageIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshall an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.461.1 Detailed Description

Marshaling code for Open Wire Format for **MessageIdMarshaller** (p.2150). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.461.2 Constructor & Destructor Documentation

6.461.2.1 activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller::MessageIdMarshaller () [inline]

6.461.2.2 virtual activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller::~~MessageIdMarshaller () [inline, virtual]

6.461.3 Member Function Documentation

6.461.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1331).

6.461.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1336).

6.461.3.3 virtual void activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1342).

6.461.3.4 virtual void activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1348).

6.461.3.5 virtual int activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1354).

```
6.461.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::MessageIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1360).

```
6.461.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::MessageIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1366).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/MessageIdMarshaller.h`

6.462 activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller Class Reference

Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2154).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/MessageIdMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller**:

Public Member Functions

- **MessageIdMarshaller** ()
- virtual **~MessageIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.462.1 Detailed Description

Marshaling code for Open Wire Format for **MessageIdMarshaller** (p.2154). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.462.2 Constructor & Destructor Documentation

6.462.2.1 `activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller::MessageIdMarshaller () [inline]`

6.462.2.2 `virtual activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller::~~MessageIdMarshaller () [inline, virtual]`

6.462.3 Member Function Documentation

6.462.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1331).

6.462.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1336).

6.462.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1342).

```
6.462.3.4 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::MessageIdMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1348).

```
6.462.3.5 virtual int ac-
tivemq::wireformat::openwire::marshal::v3::MessageIdMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, utils::BooleanStream * bs ) throw (
decaf::io::IOException ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1354).

6.462.3.6 `virtual void activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1360).

6.462.3.7 `virtual void activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1366).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/MessageIdMarshaller.h`

6.463 `activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller` Class Reference

Marshaling code for Open Wire Format for `MessageIdMarshaller` (p. 2157).


```
#include <src/main/activemq/wireformat/openwire/marshal/v5/MessageIdMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller:

Public Member Functions

- **MessageIdMarshaller** ()
- virtual **~MessageIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.463.1 Detailed Description

Marshaling code for Open Wire Format for **MessageIdMarshaller** (p.2157). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.463.2 Constructor & Destructor Documentation

6.463.2.1 `activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller::MessageIdMarshaller () [inline]`

6.463.2.2 `virtual activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller::~~MessageIdMarshaller () [inline, virtual]`

6.463.3 Member Function Documentation

6.463.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.463.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.463.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1342).

6.463.3.4 virtual void activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1348).

6.463.3.5 virtual int activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1354).

6.463.3.6 virtual void activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1360).

6.463.3.7 virtual void **activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller::tightUnmarshal**
 (**OpenWireFormat** * *wireFormat*, **commands::DataStructure**
 * *dataStructure*, **decaf::io::DataInputStream** * *dataIn*,
utils::BooleanStream * *bs*) throw (**decaf::io::IOException**) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1366).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/MessageIdMarshaller.h`

6.464 **activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller** Class Reference

Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2161).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/MessageIdMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller**:

Public Member Functions

- **MessageIdMarshaller** ()
- virtual **~MessageIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.464.1 Detailed Description

Marshaling code for Open Wire Format for **MessageIdMarshaller** (p.2161). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.464.2 Constructor & Destructor Documentation

6.464.2.1 `activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller::MessageIdMarshaller () [inline]`

6.464.2.2 `virtual activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller::~~MessageIdMarshaller () [inline, virtual]`

6.464.3 Member Function Documentation

6.464.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.464.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.464.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1342).

6.464.3.4 virtual void activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1348).

6.464.3.5 virtual int activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1354).

6.464.3.6 virtual void activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions

- IOException* if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1360).

6.464.3.7 virtual void **activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller::tightUnmarshal**
 (**OpenWireFormat** * *wireFormat*, **commands::DataStructure**
 * *dataStructure*, **decaf::io::DataInputStream** * *dataIn*,
utils::BooleanStream * *bs*) throw (**decaf::io::IOException**) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1366).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/MessageIdMarshaller.h`

6.465 cms::MessageListener Class Reference

A **MessageListener** (p. 2165) object is used to receive asynchronously delivered messages.

```
#include <src/main/cms/MessageListener.h>
```

Public Member Functions

- virtual **~MessageListener** ()
- virtual void **onMessage** (const **Message** *message)=0
*Called asynchronously when a new message is received, the message reference can be to any of the **Message** (p. 2036) types.*

6.465.1 Detailed Description

A `MessageListener` (p. 2165) object is used to receive asynchronously delivered messages.

Since

1.0

6.465.2 Constructor & Destructor Documentation

6.465.2.1 `virtual cms::MessageListener::~~MessageListener () [inline, virtual]`

6.465.3 Member Function Documentation

6.465.3.1 `virtual void cms::MessageListener::onMessage (const Message * message) [pure virtual]`

Called asynchronously when a new message is received, the message reference can be to any of the `Message` (p. 2036) types.

a dynamic cast is used to find out what type of message this is. The lifetime of this object is only guaranteed to be for life of the `onMessage` function after this call-back returns the message may no longer exists. Users should copy the data or clone the message if they wish to retain information that was contained in this `Message` (p. 2036).

It is considered a programming error for this method to throw an exception.

Parameters

message `Message` (p. 2036) object {const} pointer recipient does not own.

The documentation for this class was generated from the following file:

- `src/main/cms/MessageListener.h`

6.466 activemq::wireformat::openwire::marshal::v2::MessageMarshaller Class Reference

Marshaling code for Open Wire Format for `MessageMarshaller` (p. 2166).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/MessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::MessageMarshaller`:

Public Member Functions

- `MessageMarshaller ()`
- `virtual ~MessageMarshaller ()`
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.466.1 Detailed Description

Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2166). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.466.2 Constructor & Destructor Documentation

6.466.2.1 activemq::wireformat::openwire::marshal::v2::MessageMarshaller::MessageMarshaller () [inline]

6.466.2.2 virtual activemq::wireformat::openwire::marshal::v2::MessageMarshaller::~~MessageMarshaller () [inline, virtual]

6.466.3 Member Function Documentation

6.466.3.1 virtual void activemq::wireformat::openwire::marshal::v2::MessageMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 631).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller` (p. 164), `activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller` (p. 200), `activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller` (p. 302), `activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller` (p. 325), `activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller` (p. 365), `activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller` (p. 453), and `activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller` (p. 547).

```
6.466.3.2 virtual void ac-
      tivemq::wireformat::openwire::marshal::v2::MessageMarshaller::looseUnmarshal
      ( OpenWireFormat * wireFormat, commands::DataStructure *
      dataStructure, decaf::io::DataInputStream * dataIn ) throw (
      decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 632).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller` (p. 165), `activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller` (p. 200), `activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller` (p. 303), `activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller` (p. 325), `activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller` (p. 365), `activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller` (p. 454), and `activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller` (p. 548).

```

6.466.3.3 virtual int ac-
tivismq::wireformat::openwire::marshal::v2::MessageMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, utils::BooleanStream * bs ) throw (
decaf::io::IOException ) [virtual]

```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 633).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller` (p. 165), `activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller` (p. 200), `activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller` (p. 303), `activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller` (p. 325), `activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller` (p. 365), `activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller` (p. 454), and `activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller` (p. 548).

```

6.466.3.4 virtual void ac-
tivismq::wireformat::openwire::marshal::v2::MessageMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 634).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller` (p. 165), `activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller` (p. 201), `activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller` (p. 303), `activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller` (p. 326), `activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller` (p. 366), `activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller` (p. 454), and `activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller` (p. 548).

```
6.466.3.5 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::MessageMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 635).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller` (p. 166), `activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller` (p. 201), `activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller` (p. 304), `activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller` (p. 326), `activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller` (p. 366), `activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller` (p. 455), and `activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller` (p. 549).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/MessageMarshaller.h`

6.467 activemq::wireformat::openwire::marshal::v4::MessageMarshaller Class Reference

Marshaling code for Open Wire Format for `MessageMarshaller` (p. 2170).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/MessageMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v4::MessageMarshaller:

Public Member Functions

- **MessageMarshaller** ()
- virtual **~MessageMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

6.467.1 Detailed Description

Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2170). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.467.2 Constructor & Destructor Documentation

6.467.2.1 `activemq::wireformat::openwire::marshal::v4::MessageMarshaller::MessageMarshaller ()` [inline]

6.467.2.2 `virtual activemq::wireformat::openwire::marshal::v4::MessageMarshaller::~~MessageMarshaller ()` [inline, virtual]

6.467.3 Member Function Documentation

6.467.3.1 `virtual void activemq::wireformat::openwire::marshal::v4::MessageMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 618).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller` (p. 156), `activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller` (p. 192), `activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller` (p. 294), `activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller` (p. 317), `activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller` (p. 357), `activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller` (p. 445), and `activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller` (p. 539).

6.467.3.2 `virtual void activemq::wireformat::openwire::marshal::v4::MessageMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 619).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller` (p. 157), `activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller` (p. 192), `activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller` (p. 295), `activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller` (p. 317), `activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller` (p. 357), `activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller` (p. 446), and `activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller` (p. 540).

```
6.467.3.3 virtual int ac-
      tivemq::wireformat::openwire::marshal::v4::MessageMarshaller::tightMarshal1
      ( OpenWireFormat * wireFormat, commands::DataStructure
        * dataStructure, utils::BooleanStream * bs ) throw (
        decaf::io::IOException ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 620).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller` (p. 157), `activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller` (p. 192), `activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller` (p. 295), `activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller` (p. 317), `activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller` (p. 357), `activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller` (p. 446), and `activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller` (p. 540).

6.467.3.4 virtual void activemq::wireformat::openwire::marshal::v4::MessageMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 621).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller** (p. 157), **activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller** (p. 193), **activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller** (p. 295), **activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller** (p. 318), **activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller** (p. 358), **activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller** (p. 446), and **activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller** (p. 540).

6.467.3.5 virtual void activemq::wireformat::openwire::marshal::v4::MessageMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 622).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller` (p. 158), `activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller` (p. 193), `activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller` (p. 296), `activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller` (p. 318), `activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller` (p. 358), `activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller` (p. 447), and `activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller` (p. 541).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/MessageMarshaller.h`

6.468 `activemq::wireformat::openwire::marshal::v3::MessageMarshaller` Class Reference

Marshaling code for Open Wire Format for `MessageMarshaller` (p. 2175).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/MessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::MessageMarshaller`:

Public Member Functions

- `MessageMarshaller ()`
- `virtual ~MessageMarshaller ()`
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write a object instance to data output stream.
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)`
Write a object instance to data output stream.

6.468.1 Detailed Description

Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2175). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.468.2 Constructor & Destructor Documentation

6.468.2.1 `activemq::wireformat::openwire::marshal::v3::MessageMarshaller::MessageMarshaller ()` [inline]

6.468.2.2 `virtual`
`activemq::wireformat::openwire::marshal::v3::MessageMarshaller::~~MessageMarshaller ()` [inline, virtual]

6.468.3 Member Function Documentation

6.468.3.1 `virtual void` `activemq::wireformat::openwire::marshal::v3::MessageMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut)` throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 604).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller` (p. 148), `activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller` (p. 184), `activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller` (p. 286), `activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller` (p. 309), `activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller` (p. 349), `activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller` (p. 437), and `activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller` (p. 531).

6.468.3.2 `virtual void` `activemq::wireformat::openwire::marshal::v3::MessageMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn)` throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 605).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller` (p. 149), `activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller` (p. 184), `activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller` (p. 287), `activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller` (p. 309), `activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller` (p. 349), `activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller` (p. 438), and `activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller` (p. 532).

```
6.468.3.3  virtual int ac-
            tivemq::wireformat::openwire::marshal::v3::MessageMarshaller::tightMarshal1
            ( OpenWireFormat * wireFormat, commands::DataStructure
              * dataStructure, utils::BooleanStream * bs ) throw (
              decaf::io::IOException ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 606).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller` (p. 149), `activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller` (p. 184), `activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller` (p. 287), `activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller` (p. 309), `activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller` (p. 349), `activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller` (p. 438), and `activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller` (p. 532).

6.468.3.4 virtual void activemq::wireformat::openwire::marshal::v3::MessageMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 608).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller** (p. 149), **activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller** (p. 185), **activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller** (p. 287), **activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller** (p. 310), **activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller** (p. 350), **activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller** (p. 439), and **activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller** (p. 532).

6.468.3.5 virtual void activemq::wireformat::openwire::marshal::v3::MessageMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 609).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller` (p. 150), `activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller` (p. 185), `activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller` (p. 288), `activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller` (p. 310), `activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller` (p. 350), `activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller` (p. 439), and `activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller` (p. 533).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/MessageMarshaller.h`

6.469 `activemq::wireformat::openwire::marshal::v5::MessageMarshaller` Class Reference

Marshaling code for Open Wire Format for `MessageMarshaller` (p. 2179).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/MessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v5::MessageMarshaller`:

Public Member Functions

- `MessageMarshaller ()`
- `virtual ~MessageMarshaller ()`
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write a object instance to data output stream.
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)`
Write a object instance to data output stream.

6.469.1 Detailed Description

Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2179). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.469.2 Constructor & Destructor Documentation

6.469.2.1 `activemq::wireformat::openwire::marshal::v5::MessageMarshaller::MessageMarshaller ()` [inline]

6.469.2.2 `virtual activemq::wireformat::openwire::marshal::v5::MessageMarshaller::~~MessageMarshaller ()` [inline, virtual]

6.469.3 Member Function Documentation

6.469.3.1 `virtual void activemq::wireformat::openwire::marshal::v5::MessageMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 624).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller` (p. 160), `activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller` (p. 196), `activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller` (p. 298), `activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller` (p. 321), `activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller` (p. 361), `activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller` (p. 449), and `activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller` (p. 543).

6.469.3.2 `virtual void activemq::wireformat::openwire::marshal::v5::MessageMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 625).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller` (p. 161), `activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller` (p. 196), `activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller` (p. 299), `activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller` (p. 321), `activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller` (p. 361), `activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller` (p. 450), and `activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller` (p. 544).

```
6.469.3.3 virtual int ac-
          tivemq::wireformat::openwire::marshal::v5::MessageMarshaller::tightMarshal1
          ( OpenWireFormat * wireFormat, commands::DataStructure
            * dataStructure, utils::BooleanStream * bs ) throw (
            decaf::io::IOException ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 626).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller` (p. 161), `activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller` (p. 196), `activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller` (p. 299), `activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller` (p. 321), `activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller` (p. 361), `activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller` (p. 450), and `activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller` (p. 544).


```

6.469.3.4 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::MessageMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 628).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller** (p. 161), **activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller** (p. 197), **activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller** (p. 299), **activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller** (p. 322), **activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller** (p. 362), **activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller** (p. 450), and **activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller** (p. 544).

```

6.469.3.5 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::MessageMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]

```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 629).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller` (p. 162), `activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller` (p. 197), `activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller` (p. 300), `activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller` (p. 322), `activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller` (p. 362), `activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller` (p. 451), and `activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller` (p. 545).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/MessageMarshaller.h`

6.470 `activemq::wireformat::openwire::marshal::v1::MessageMarshaller` Class Reference

Marshaling code for Open Wire Format for `MessageMarshaller` (p. 2183).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/MessageMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v1::MessageMarshaller`:

Public Member Functions

- `MessageMarshaller ()`
- `virtual ~MessageMarshaller ()`
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write a object instance to data output stream.
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)`
Write a object instance to data output stream.

6.470.1 Detailed Description

Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2183). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.470.2 Constructor & Destructor Documentation

6.470.2.1 `activemq::wireformat::openwire::marshal::v1::MessageMarshaller::MessageMarshaller ()` [inline]

6.470.2.2 `virtual activemq::wireformat::openwire::marshal::v1::MessageMarshaller::~~MessageMarshaller ()` [inline, virtual]

6.470.3 Member Function Documentation

6.470.3.1 `virtual void activemq::wireformat::openwire::marshal::v1::MessageMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 611).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller` (p. 152), `activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller` (p. 188), `activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller` (p. 290), `activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller` (p. 313), `activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller` (p. 353), `activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller` (p. 441), and `activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller` (p. 535).

6.470.3.2 `virtual void activemq::wireformat::openwire::marshal::v1::MessageMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 612).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller` (p. 153), `activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller` (p. 188), `activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller` (p. 291), `activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller` (p. 313), `activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller` (p. 353), `activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller` (p. 442), and `activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller` (p. 536).

```
6.470.3.3 virtual int ac-
            tivemq::wireformat::openwire::marshal::v1::MessageMarshaller::tightMarshal1
            ( OpenWireFormat * wireFormat, commands::DataStructure
              * dataStructure, utils::BooleanStream * bs ) throw (
              decaf::io::IOException ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 613).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller` (p. 153), `activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller` (p. 188), `activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller` (p. 291), `activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller` (p. 313), `activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller` (p. 353), `activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller` (p. 442), and `activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller` (p. 536).

6.470.3.4 virtual void activemq::wireformat::openwire::marshal::v1::MessageMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 614).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller** (p. 153), **activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller** (p. 189), **activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller** (p. 291), **activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller** (p. 314), **activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller** (p. 354), **activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller** (p. 442), and **activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller** (p. 536).

6.470.3.5 virtual void activemq::wireformat::openwire::marshal::v1::MessageMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 615).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller` (p. 154), `activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller` (p. 189), `activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller` (p. 292), `activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller` (p. 314), `activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller` (p. 354), `activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller` (p. 443), and `activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller` (p. 537).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/MessageMarshaller.h`

6.471 cms::MessageNotReadableException Class Reference

This exception must be thrown when a CMS client attempts to read a write-only message.

```
#include <src/main/cms/MessageNotReadableException.h>
```

Inheritance diagram for `cms::MessageNotReadableException`:

Public Member Functions

- `MessageNotReadableException () throw ()`
- `MessageNotReadableException (const MessageNotReadableException &ex) throw ()`
- `MessageNotReadableException (const std::string &message, const std::exception *cause) throw ()`
- `MessageNotReadableException (const std::string &message, const std::exception *cause, const std::vector< std::pair< std::string, int > > &stackTrace) throw ()`
- `virtual ~MessageNotReadableException () throw ()`

6.471.1 Detailed Description

This exception must be thrown when a CMS client attempts to read a write-only message.

Since

1.3

6.471.2 Constructor & Destructor Documentation

- 6.471.2.1 `cms::MessageNotReadableException::MessageNotReadableException () throw ()`
- 6.471.2.2 `cms::MessageNotReadableException::MessageNotReadableException (const MessageNotReadableException & ex) throw ()`
- 6.471.2.3 `cms::MessageNotReadableException::MessageNotReadableException (const std::string & message, const std::exception * cause) throw ()`
- 6.471.2.4 `cms::MessageNotReadableException::MessageNotReadableException (const std::string & message, const std::exception * cause, const std::vector< std::pair< std::string, int > > & stackTrace) throw ()`
- 6.471.2.5 `virtual cms::MessageNotReadableException::~~MessageNotReadableException () throw () [virtual]`

The documentation for this class was generated from the following file:

- `src/main/cms/MessageNotReadableException.h`

6.472 cms::MessageNotWriteableException Class Reference

This exception must be thrown when a CMS client attempts to write to a read-only message.

```
#include <src/main/cms/MessageNotWriteableException.h>
```

Inheritance diagram for cms::MessageNotWriteableException:

Public Member Functions

- `MessageNotWriteableException () throw ()`
- `MessageNotWriteableException (const MessageNotWriteableException &ex) throw ()`
- `MessageNotWriteableException (const std::string &message, const std::exception *cause) throw ()`
- `MessageNotWriteableException (const std::string &message, const std::exception *cause, const std::vector< std::pair< std::string, int > > &stackTrace) throw ()`
- `virtual ~MessageNotWriteableException () throw ()`

6.472.1 Detailed Description

This exception must be thrown when a CMS client attempts to write to a read-only message.

Since

1.3

6.472.2 Constructor & Destructor Documentation

- 6.472.2.1 `cms::MessageNotWriteableException::MessageNotWriteableException () throw ()`
- 6.472.2.2 `cms::MessageNotWriteableException::MessageNotWriteableException (const MessageNotWriteableException & ex) throw ()`
- 6.472.2.3 `cms::MessageNotWriteableException::MessageNotWriteableException (const std::string & message, const std::exception * cause) throw ()`
- 6.472.2.4 `cms::MessageNotWriteableException::MessageNotWriteableException (const std::string & message, const std::exception * cause, const std::vector< std::pair< std::string, int > > & stackTrace) throw ()`
- 6.472.2.5 `virtual cms::MessageNotWriteableException::~~MessageNotWriteableException () throw () [virtual]`

The documentation for this class was generated from the following file:

- `src/main/cms/MessageNotWriteableException.h`

6.473 cms::MessageProducer Class Reference

A client uses a `MessageProducer` (p. 2189) object to send messages to a `Destination` (p. 1387).

`#include <src/main/cms/MessageProducer.h>`

Inheritance diagram for `cms::MessageProducer`:

Public Member Functions

- `virtual ~MessageProducer ()`
- `virtual void send (Message *message)=0 throw (cms::CMSEException, cms::MessageFormatException, cms::InvalidDestinationException, cms::UnsupportedOperationException)`
Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.
- `virtual void send (Message *message, int deliveryMode, int priority, long long timeToLive)=0 throw (cms::CMSEException, cms::MessageFormatException, cms::InvalidDestinationException, cms::UnsupportedOperationException)`
Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.
- `virtual void send (const Destination *destination, Message *message)=0 throw (cms::CMSEException, cms::MessageFormatException, cms::InvalidDestinationException, cms::UnsupportedOperationException)`

Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.

- virtual void **send** (const **Destination** *destination, **Message** *message, int deliveryMode, int priority, long long timeToLive)=0 throw (cms::CMSEException, cms::MessageFormatException, cms::InvalidDestinationException, cms::UnsupportedOperationException)

Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.

- virtual void **setDeliveryMode** (int mode)=0 throw (CMSEException)

Sets the delivery mode for this Producer.

- virtual int **getDeliveryMode** () const =0 throw (CMSEException)

Gets the delivery mode for this Producer.

- virtual void **setDisableMessageID** (bool value)=0 throw (CMSEException)

*Sets if **Message** (p. 2036) Ids are disabled for this Producer.*

- virtual bool **getDisableMessageID** () const =0 throw (CMSEException)

*Gets if **Message** (p. 2036) Ids are disabled for this Producer.*

- virtual void **setDisableMessageTimeStamp** (bool value)=0 throw (CMSEException)

*Sets if **Message** (p. 2036) Time Stamps are disabled for this Producer.*

- virtual bool **getDisableMessageTimeStamp** () const =0 throw (CMSEException)

*Gets if **Message** (p. 2036) Time Stamps are disabled for this Producer.*

- virtual void **setPriority** (int priority)=0 throw (CMSEException)

Sets the Priority that this Producers sends messages at.

- virtual int **getPriority** () const =0 throw (CMSEException)

Gets the Priority level that this producer sends messages at.

- virtual void **setTimeToLive** (long long time)=0 throw (CMSEException)

Sets the Time to Live that this Producers sends messages with.

- virtual long long **getTimeToLive** () const =0 throw (CMSEException)

Gets the Time to Live that this producer sends messages with.

6.473.1 Detailed Description

A client uses a **MessageProducer** (p. 2189) object to send messages to a **Destination** (p. 1387). A **MessageProducer** (p. 2189) object is created by passing a **Destination** (p. 1387) object to a message-producer creation method supplied by a session.

A client also has the option of creating a message producer without supplying a destination. In this case, a **Destination** (p. 1387) must be provided with every send operation. A typical use for this kind of message producer is to send replies to requests using the request's **CMSReplyTo** destination.

A client can specify a default delivery mode, priority, and time to live for messages sent by a message producer. It can also specify the delivery mode, priority, and time to live for an individual message.

A client can specify a time-to-live value in milliseconds for each message it sends. This value defines a message expiration time that is the sum of the message's time-to-live and the GMT when it is sent (for transacted sends, this is the time the client sends the message, not the time the transaction is committed).

Since

1.0

6.473.2 Constructor & Destructor Documentation

6.473.2.1 `virtual cms::MessageProducer::~MessageProducer () [inline, virtual]`

6.473.3 Member Function Documentation

6.473.3.1 `virtual int cms::MessageProducer::getDeliveryMode () const throw (CMSEException) [pure virtual]`

Gets the delivery mode for this Producer.

Returns

The **DeliveryMode** (p. 1386)

Exceptions

CMSEException (p. 960) - if an internal error occurs.

Implemented in `activemq::cmsutil::CachedProducer` (p. 892), and `activemq::core::ActiveMQProducer` (p. 369).

6.473.3.2 `virtual bool cms::MessageProducer::getDisableMessageID () const throw (CMSEException) [pure virtual]`

Gets if **Message** (p. 2036) Ids are disabled for this Producer.

Returns

boolean indicating enable / disable (true / false)

Exceptions

CMSEException (p. 960) - if an internal error occurs.

Implemented in `activemq::cmsutil::CachedProducer` (p. 893), and `activemq::core::ActiveMQProducer` (p. 369).

6.473.3.3 `virtual bool cms::MessageProducer::getDisableMessageTimeStamp ()
const throw (CMSEException) [pure virtual]`

Gets if `Message` (p. 2036) Time Stamps are disabled for this Producer.

Returns

boolean indicating enable / disable (true / false)

Exceptions

CMSEException (p. 960) - if an internal error occurs.

Implemented in `activemq::cmsutil::CachedProducer` (p. 893), and `activemq::core::ActiveMQProducer` (p. 370).

6.473.3.4 `virtual int cms::MessageProducer::getPriority () const throw (CMSEException) [pure virtual]`

Gets the Priority level that this producer sends messages at.

Returns

int based priority level

Exceptions

CMSEException (p. 960) - if an internal error occurs.

Implemented in `activemq::cmsutil::CachedProducer` (p. 893), and `activemq::core::ActiveMQProducer` (p. 370).

6.473.3.5 `virtual long long cms::MessageProducer::getTimeToLive () const
throw (CMSEException) [pure virtual]`

Gets the Time to Live that this producer sends messages with.

Returns

Time to live value in milliseconds

Exceptions

CMSEException (p. 960) - if an internal error occurs.

Implemented in `activemq::cmsutil::CachedProducer` (p. 894), and `activemq::core::ActiveMQProducer` (p. 371).

6.473.3.6 `virtual void cms::MessageProducer::send (Message * message,
int deliveryMode, int priority, long long timeToLive)
throw (cms::CMSEException, cms::MessageFormatException,
cms::InvalidDestinationException, cms::UnsupportedOperationException) [pure virtual]`

Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.

Parameters

message The message to be sent.

deliveryMode The delivery mode to be used.

priority The priority for this message.

timeToLive The time to live value for this message in milliseconds.

Exceptions

CMSEException (p. 960) - if an internal error occurs while sending the message.

MessageFormatException (p. 2141) - if an Invalid **Message** (p. 2036) is given.

InvalidDestinationException (p. 1697) - if a client uses this method with a **MessageProducer** (p. 2189) with an invalid destination.

UnsupportedOperationException (p. 3093) - if a client uses this method with a **MessageProducer** (p. 2189) that did not specify a destination at creation time.

Implemented in **activemq::cmsutil::CachedProducer** (p. 895), and **activemq::core::ActiveMQProducer** (p. 372).

6.473.3.7 virtual void cms::MessageProducer::send (const Destination * destination, Message * message, int deliveryMode, int priority, long long timeToLive) throw (cms::CMSEException, cms::MessageFormatException, cms::InvalidDestinationException, cms::UnsupportedOperationException) [pure virtual]

Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.

Parameters

destination The destination on which to send the message

message The message to be sent.

deliveryMode The delivery mode to be used.

priority The priority for this message.

timeToLive The time to live value for this message in milliseconds.

Exceptions

CMSEException (p. 960) - if an internal error occurs while sending the message.

MessageFormatException (p. 2141) - if an Invalid **Message** (p. 2036) is given.

InvalidDestinationException (p. 1697) - if a client uses this method with a **MessageProducer** (p. 2189) with an invalid destination.

UnsupportedOperationException (p. 3093) - if a client uses this method with a **MessageProducer** (p. 2189) that did not specify a destination at creation time.

Implemented in **activemq::cmsutil::CachedProducer** (p. 895), and **activemq::core::ActiveMQProducer** (p. 371).

6.473.3.8 virtual void cms::MessageProducer::send (Message * *message*) throw (cms::CMSException, cms::MessageFormatException, cms::InvalidDestinationException, cms::UnsupportedOperationException) [pure virtual]

Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.

Uses default values for deliveryMode, priority, and time to live.

Parameters

message The message to be sent.

Exceptions

CMSException (p. 960) - if an internal error occurs while sending the message.

MessageFormatException (p. 2141) - if an Invalid **Message** (p. 2036) is given.

InvalidDestinationException (p. 1697) - if a client uses this method with a **MessageProducer** (p. 2189) with an invalid destination.

UnsupportedOperationException (p. 3093) - if a client uses this method with a **MessageProducer** (p. 2189) that did not specify a destination at creation time.

Implemented in **activemq::cmsutil::CachedProducer** (p. 894), and **activemq::core::ActiveMQProducer** (p. 372).

6.473.3.9 virtual void cms::MessageProducer::send (const Destination * *destination*, Message * *message*) throw (cms::CMSException, cms::MessageFormatException, cms::InvalidDestinationException, cms::UnsupportedOperationException) [pure virtual]

Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.

Uses default values for deliveryMode, priority, and time to live.

Parameters

destination The destination on which to send the message

message the message to be sent.

Exceptions

CMSException (p. 960) - if an internal error occurs while sending the message.

MessageFormatException (p. 2141) - if an Invalid **Message** (p. 2036) is given.

InvalidDestinationException (p. 1697) - if a client uses this method with a **MessageProducer** (p. 2189) with an invalid destination.

UnsupportedOperationException (p. 3093) - if a client uses this method with a **MessageProducer** (p. 2189) that did not specify a destination at creation time.

Implemented in **activemq::cmsutil::CachedProducer** (p. 894), and **activemq::core::ActiveMQProducer** (p. 373).

6.473.3.10 `virtual void cms::MessageProducer::setDeliveryMode (int mode)
throw (CMSEException)` [pure virtual]

Sets the delivery mode for this Producer.

Parameters

mode The **DeliveryMode** (p. 1386)

Exceptions

CMSEException (p. 960) - if an internal error occurs.

Implemented in **activemq::cmsutil::CachedProducer** (p. 896), and **activemq::core::ActiveMQProducer** (p. 373).

6.473.3.11 `virtual void cms::MessageProducer::setDisableMessageID (bool value)
throw (CMSEException)` [pure virtual]

Sets if **Message** (p. 2036) Ids are disabled for this Producer.

Parameters

value boolean indicating enable / disable (true / false)

Exceptions

CMSEException (p. 960) - if an internal error occurs.

Implemented in **activemq::cmsutil::CachedProducer** (p. 896), and **activemq::core::ActiveMQProducer** (p. 373).

6.473.3.12 `virtual void cms::MessageProducer::setDisableMessageTimeStamp (bool value)
throw (CMSEException)` [pure virtual]

Sets if **Message** (p. 2036) Time Stamps are disabled for this Producer.

Parameters

value - boolean indicating enable / disable (true / false)

Exceptions

CMSEException (p. 960) - if an internal error occurs.

Implemented in **activemq::cmsutil::CachedProducer** (p. 896), and **activemq::core::ActiveMQProducer** (p. 374).

6.473.3.13 `virtual void cms::MessageProducer::setPriority (int priority) throw
(CMSEException)` [pure virtual]

Sets the Priority that this Producers sends messages at.

Parameters

priority int value for Priority level

Exceptions

CMSEException (p. 960) - if an internal error occurs.

Implemented in **activemq::cmsutil::CachedProducer** (p. 897), and **activemq::core::ActiveMQProducer** (p. 374).

6.473.3.14 virtual void cms::MessageProducer::setTimeToLive (long long *time*) throw (CMSEException) [pure virtual]

Sets the Time to Live that this Producers sends messages with.

This value will be used if the time to live is not specified via the send method.

Parameters

time default time to live value in milliseconds

Exceptions

CMSEException (p. 960) - if an internal error occurs.

Implemented in **activemq::cmsutil::CachedProducer** (p. 897), and **activemq::core::ActiveMQProducer** (p. 374).

The documentation for this class was generated from the following file:

- src/main/cms/MessageProducer.h

6.474 activemq::wireformat::openwire::utils::MessagePropertyInterceptor Class Reference

Used the base ActiveMQMessage class to intercept calls to get and set properties in order to capture the calls that use the reserved JMS properties and get and set them in the OpenWire Message properties.

```
#include <src/main/activemq/wireformat/openwire/utils/MessagePropertyInterceptor.h>
```

Public Member Functions

- **MessagePropertyInterceptor** (commands::Message *message, util::PrimitiveMap *properties) throw (decaf::lang::exceptions::NullPointerException)

Constructor, accepts the Message that will be used to store JMS reserved property values, and the PrimitiveMap to get and set the rest to.

- virtual ~**MessagePropertyInterceptor** ()
- virtual bool **getBooleanProperty** (const std::string &name) const

Gets a boolean property.

- virtual unsigned char **getByteProperty** (const std::string &name) const
Gets a byte property.
- virtual double **getDoubleProperty** (const std::string &name) const
Gets a double property.
- virtual float **getFloatProperty** (const std::string &name) const
Gets a float property.
- virtual int **getIntProperty** (const std::string &name) const
Gets a int property.
- virtual long long **getLongProperty** (const std::string &name) const
Gets a long property.
- virtual short **getShortProperty** (const std::string &name) const
Gets a short property.
- virtual std::string **getStringProperty** (const std::string &name) const
Gets a string property.
- virtual void **setBooleanProperty** (const std::string &name, bool value)
Sets a boolean property.
- virtual void **setByteProperty** (const std::string &name, unsigned char value)
Sets a byte property.
- virtual void **setDoubleProperty** (const std::string &name, double value)
Sets a double property.
- virtual void **setFloatProperty** (const std::string &name, float value)
Sets a float property.
- virtual void **setIntProperty** (const std::string &name, int value)
Sets a int property.
- virtual void **setLongProperty** (const std::string &name, long long value)
Sets a long property.
- virtual void **setShortProperty** (const std::string &name, short value)
Sets a short property.
- virtual void **setStringProperty** (const std::string &name, const std::string &value)
Sets a string property.

6.474.1 Detailed Description

Used the base ActiveMQMessage class to intercept calls to get and set properties in order to capture the calls that use the reserved JMS properties and get and set them in the OpenWire Message properties. Currently the only properties that are intercepted and handled are:

Name | Conversion Supported ----- JMSXDeliveryCount |
Int, Long, String JMSXGroupID | String JMSXGroupSeq | Int, Long, String

6.474.2 Constructor & Destructor Documentation

6.474.2.1 `activemq::wireformat::openwire::utils::MessagePropertyInterceptor::MessagePropertyInterceptor (commands::Message * message, util::PrimitiveMap * properties)
throw (decaf::lang::exceptions::NullPointerException)`

Constructor, accepts the Message that will be used to store JMS reserved property values, and the PrimitiveMap to get and set the rest to.

Parameters

message - The Message to store reserved property data in
properties - The PrimitiveMap to store the rest of the properties in.

Exceptions

NullPointerException if either param is NULL

6.474.2.2 `virtual
activemq::wireformat::openwire::utils::MessagePropertyInterceptor::~MessagePropertyInterceptor () [virtual]`

6.474.3 Member Function Documentation

6.474.3.1 `virtual bool activemq::wireformat::openwire::utils::MessagePropertyInterceptor::getBooleanProperty (const std::string & name) const [virtual]`

Gets a boolean property.

Parameters

name The name of the property to retrieve.

Returns

The value for the named property.

6.474.3.2 `virtual unsigned char activemq::wireformat::openwire::utils::MessagePropertyInterceptor::getByteProperty (const std::string & name) const [virtual]`

Gets a byte property.

Parameters

name The name of the property to retrieve.

Returns

The value for the named property.

6.474.3.3 `virtual double activemq::wireformat::openwire::utils::MessagePropertyInterceptor::getDoubleProperty (const std::string & name) const [virtual]`

Gets a double property.

Parameters

name The name of the property to retrieve.

Returns

The value for the named property.

6.474.3.4 `virtual float activemq::wireformat::openwire::utils::MessagePropertyInterceptor::getFloatProperty (const std::string & name) const [virtual]`

Gets a float property.

Parameters

name The name of the property to retrieve.

Returns

The value for the named property.

6.474.3.5 `virtual int activemq::wireformat::openwire::utils::MessagePropertyInterceptor::getIntProperty (const std::string & name) const [virtual]`

Gets a int property.

Parameters

name The name of the property to retrieve.

Returns

The value for the named property.

6.474.3.6 virtual long long activemq::wireformat::openwire::utils::MessagePropertyInterceptor::getLongProperty
(const std::string & *name*) const [virtual]

Gets a long property.

Parameters

name The name of the property to retrieve.

Returns

The value for the named property.

6.474.3.7 virtual short short activemq::wireformat::openwire::utils::MessagePropertyInterceptor::getShortProperty
(const std::string & *name*) const [virtual]

Gets a short property.

Parameters

name The name of the property to retrieve.

Returns

The value for the named property.

6.474.3.8 virtual std::string string activemq::wireformat::openwire::utils::MessagePropertyInterceptor::getStringProperty
(const std::string & *name*) const [virtual]

Gets a string property.

Parameters

name The name of the property to retrieve.

Returns

The value for the named property.

6.474.3.9 virtual void void activemq::wireformat::openwire::utils::MessagePropertyInterceptor::setBooleanProperty
(const std::string & *name*, bool *value*) [virtual]

Sets a boolean property.

Parameters

name The name of the property to retrieve.

value The value for the named property.

6.474.3.10 `virtual void activemq::wireformat::openwire::utils::MessagePropertyInterceptor::setByteProperty(const std::string & name, unsigned char value) [virtual]`

Sets a byte property.

Parameters

name The name of the property to retrieve.

value The value for the named property.

6.474.3.11 `virtual void activemq::wireformat::openwire::utils::MessagePropertyInterceptor::setDoubleProperty(const std::string & name, double value) [virtual]`

Sets a double property.

Parameters

name The name of the property to retrieve.

value The value for the named property.

6.474.3.12 `virtual void activemq::wireformat::openwire::utils::MessagePropertyInterceptor::setFloatProperty(const std::string & name, float value) [virtual]`

Sets a float property.

Parameters

name The name of the property to retrieve.

value The value for the named property.

6.474.3.13 `virtual void activemq::wireformat::openwire::utils::MessagePropertyInterceptor::setIntProperty(const std::string & name, int value) [virtual]`

Sets a int property.

Parameters

name The name of the property to retrieve.

value The value for the named property.

6.474.3.14 `virtual void activemq::wireformat::openwire::utils::MessagePropertyInterceptor::setLongProperty(const std::string & name, long long value) [virtual]`

Sets a long property.

Parameters

name The name of the property to retrieve.

value The value for the named property.

6.474.3.15 `virtual void activemq::wireformat::openwire::utils::MessagePropertyInterceptor::setShortProperty (const std::string & name, short value) [virtual]`

Sets a short property.

Parameters

name The name of the property to retrieve.

value The value for the named property.

6.474.3.16 `virtual void activemq::wireformat::openwire::utils::MessagePropertyInterceptor::setStringProperty (const std::string & name, const std::string & value) [virtual]`

Sets a string property.

Parameters

name The name of the property to retrieve.

value The value for the named property.

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/utils/MessagePropertyInterceptor.h`

6.475 activemq::commands::MessagePull Class Reference

```
#include <src/main/activemq/commands/MessagePull.h>
```

Inheritance diagram for activemq::commands::MessagePull:

Public Member Functions

- `MessagePull ()`
- `virtual ~MessagePull ()`
- `virtual unsigned char getDataStructureType () const`
Get the unique identifier that this object and its own Marshaler share.
- `virtual MessagePull * cloneDataStructure () const`
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1372) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **ConsumerId** > & **getConsumerId** () const
- virtual **Pointer**< **ConsumerId** > & **getConsumerId** ()
- virtual void **setConsumerId** (const **Pointer**< **ConsumerId** > &consumerId)
- virtual const **Pointer**< **ActiveMQDestination** > & **getDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &destination)
- virtual long long **getTimeout** () const
- virtual void **setTimeout** (long long timeout)
- virtual const std::string & **getCorrelationId** () const
- virtual std::string & **getCorrelationId** ()
- virtual void **setCorrelationId** (const std::string &correlationId)
- virtual const **Pointer**< **MessageId** > & **getMessageId** () const
- virtual **Pointer**< **MessageId** > & **getMessageId** ()
- virtual void **setMessageId** (const **Pointer**< **MessageId** > &messageId)
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor *visitor)
 throw (exceptions::ActiveMQException)
Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_MESSAGEPULL** = 20

Protected Member Functions

- **MessagePull** (const **MessagePull** &)
- **MessagePull** & **operator=** (const **MessagePull** &)

Protected Attributes

- **Pointer**< **ConsumerId** > **consumerId**
- **Pointer**< **ActiveMQDestination** > **destination**
- long long **timeout**
- std::string **correlationId**
- **Pointer**< **MessageId** > **messageId**

6.475.1 Constructor & Destructor Documentation

6.475.1.1 `activemq::commands::MessagePull::MessagePull (const MessagePull &) [inline, protected]`

6.475.1.2 `activemq::commands::MessagePull::MessagePull ()`

6.475.1.3 `virtual activemq::commands::MessagePull::~MessagePull () [virtual]`

6.475.2 Member Function Documentation

6.475.2.1 `virtual MessagePull* activemq::commands::MessagePull::cloneDataStructure () const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1373).

6.475.2.2 `virtual void activemq::commands::MessagePull::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

Reimplemented from `activemq::commands::BaseCommand` (p. 598).

6.475.2.3 `virtual bool activemq::commands::MessagePull::equals (const DataStructure * value) const [virtual]`

Compares the `DataStructure` (p. 1372) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 598).

- 6.475.2.4 `virtual Pointer<ConsumerId>& activemq::commands::MessagePull::getConsumerId ()`
[virtual]
- 6.475.2.5 `virtual const Pointer<ConsumerId>& activemq::commands::MessagePull::getConsumerId ()`
const [virtual]
- 6.475.2.6 `virtual std::string& activemq::commands::MessagePull::getCorrelationId ()`
[virtual]
- 6.475.2.7 `virtual const std::string& activemq::commands::MessagePull::getCorrelationId ()`
const [virtual]
- 6.475.2.8 `virtual unsigned char activemq::commands::MessagePull::getDataStructureType ()`
const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataStructure** (p. 1372) type copy.

Implements **activemq::commands::DataStructure** (p. 1375).

- 6.475.2.9 virtual const Pointer<ActiveMQDestination>& activemq::commands::MessagePull::getDestination () const [virtual]
- 6.475.2.10 virtual Pointer<ActiveMQDestination>& activemq::commands::MessagePull::getDestination () [virtual]
- 6.475.2.11 virtual Pointer<MessageId>& activemq::commands::MessagePull::getMessageId () [virtual]
- 6.475.2.12 virtual const Pointer<MessageId>& activemq::commands::MessagePull::getMessageId () const [virtual]
- 6.475.2.13 virtual long long activemq::commands::MessagePull::getTimeout () const [virtual]
- 6.475.2.14 MessagePull& activemq::commands::MessagePull::operator= (const MessagePull &) [inline, protected]
- 6.475.2.15 virtual void activemq::commands::MessagePull::setConsumerId (const Pointer< ConsumerId > & *consumerId*) [virtual]
- 6.475.2.16 virtual void activemq::commands::MessagePull::setCorrelationId (const std::string & *correlationId*) [virtual]
- 6.475.2.17 virtual void activemq::commands::MessagePull::setDestination (const Pointer< ActiveMQDestination > & *destination*) [virtual]
- 6.475.2.18 virtual void activemq::commands::MessagePull::setMessageId (const Pointer< MessageId > & *messageId*) [virtual]
- 6.475.2.19 virtual void activemq::commands::MessagePull::setTimeout (long long *timeout*) [virtual]
- 6.475.2.20 virtual std::string activemq::commands::MessagePull::toString () const [virtual]

Returns a string containing the information for this **DataSet** (p.1372) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 602).

- 6.475.2.21 virtual Pointer<Command> activemq::commands::MessagePull::visit (activemq::state::CommandVisitor * *visitor*) throw (exceptions::ActiveMQException) [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

Returns

a **Response** (p. 2611) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 995).

6.475.3 Field Documentation

6.475.3.1 `Pointer<ConsumerId> activemq::commands::MessagePull::consumerId` [protected]

6.475.3.2 `std::string activemq::commands::MessagePull::correlationId` [protected]

6.475.3.3 `Pointer<ActiveMQDestination> activemq::commands::MessagePull::destination` [protected]

6.475.3.4 `const unsigned char activemq::commands::MessagePull::ID__MESSAGEPULL = 20` [static]

6.475.3.5 `Pointer<MessageId> activemq::commands::MessagePull::messageId` [protected]

6.475.3.6 `long long activemq::commands::MessagePull::timeout` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/MessagePull.h`

6.476 activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller Class Reference

Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2207).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/MessagePullMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller`:

Public Member Functions

- **MessagePullMarshaller** ()
- virtual **~MessagePullMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.

- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.476.1 Detailed Description

Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2207). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.476.2 Constructor & Destructor Documentation

- 6.476.2.1** **activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller::MessagePullMarshaller**
() [inline]
- 6.476.2.2** **virtual**
activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller::~~MessagePullMarshaller
() [inline, virtual]

6.476.3 Member Function Documentation

- 6.476.3.1** **virtual commands::DataStructure*** **activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller::createObject**
() const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.476.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller::getDataSetType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.476.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataSet * dataSet, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 631).

6.476.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataSet * dataSet, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 632).

6.476.3.5 virtual int activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 633).

6.476.3.6 virtual void activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 634).

6.476.3.7 virtual void activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 635).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/MessagePullMarshaller.h

6.477 activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller Class Reference

Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2211).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/MessagePullMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller**:

Public Member Functions

- **MessagePullMarshaller** ()
- virtual **~MessagePullMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

- virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.477.1 Detailed Description

Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2211). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.477.2 Constructor & Destructor Documentation

6.477.2.1 activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller::MessagePullMarshaller () [inline]

6.477.2.2 virtual activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller::~~MessagePullMarshaller () [inline, virtual]

6.477.3 Member Function Documentation

6.477.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1331).

6.477.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1336).

```

6.477.3.3 virtual void ac-
tivismq::wireformat::openwire::marshal::v3::MessagePullMarshaller::looseMarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 604).

```

6.477.3.4 virtual void ac-
tivismq::wireformat::openwire::marshal::v3::MessagePullMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]

```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 605).

```

6.477.3.5 virtual int ac-
tivismq::wireformat::openwire::marshal::v3::MessagePullMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, utils::BooleanStream * bs ) throw (
decaf::io::IOException ) [virtual]

```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 606).

```
6.477.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::MessagePullMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 608).

```
6.477.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::MessagePullMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 609).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/MessagePullMarshaller.h`

6.478 activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller Class Reference

Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2215).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/MessagePullMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller**:

Public Member Functions

- **MessagePullMarshaller** ()
- virtual **~MessagePullMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.478.1 Detailed Description

Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2215). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.478.2 Constructor & Destructor Documentation

6.478.2.1 **activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller::MessagePullMarshaller**
() [inline]

6.478.2.2 **virtual**
activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller::~~MessagePullMarshaller
() [inline, virtual]

6.478.3 Member Function Documentation

6.478.3.1 **virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller::createObject**
() const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1331).

6.478.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller::getDataStructureType**
() const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1336).

6.478.3.3 virtual void activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 611).

6.478.3.4 virtual void activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 612).

6.478.3.5 virtual int activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 613).

```
6.478.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::MessagePullMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 614).

```
6.478.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::MessagePullMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 615).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/MessagePullMarshaller.h`

6.479 activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller Class Reference

Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2219).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/MessagePullMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller**:

Public Member Functions

- **MessagePullMarshaller** ()
- virtual **~MessagePullMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.479.1 Detailed Description

Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2219). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.479.2 Constructor & Destructor Documentation

6.479.2.1 `activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller::MessagePullMarshaller ()` [inline]

6.479.2.2 `virtual activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller::~~MessagePullMarshaller ()` [inline, virtual]

6.479.3 Member Function Documentation

6.479.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller::createObject () const` [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1331).

6.479.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller::getDataStructureType () const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1336).

6.479.3.3 virtual void activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 624).

6.479.3.4 virtual void activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 625).

6.479.3.5 virtual int activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 626).

```
6.479.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::MessagePullMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 628).

```
6.479.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::MessagePullMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 629).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/MessagePullMarshaller.h`

6.480 **activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller** Class Reference

Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2223).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/MessagePullMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller**:

Public Member Functions

- **MessagePullMarshaller** ()
- virtual **~MessagePullMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.480.1 Detailed Description

Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2223). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.480.2 Constructor & Destructor Documentation

6.480.2.1 **activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller::MessagePullMarshaller () [inline]**

6.480.2.2 **virtual activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller::~~MessagePullMarshaller () [inline, virtual]**

6.480.3 Member Function Documentation

6.480.3.1 **virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller::createObject () const [virtual]**

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1331).

6.480.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller::getDataStructureType () const [virtual]**

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1336).

```

6.480.3.3 virtual void ac-
tivismq::wireformat::openwire::marshal::v4::MessagePullMarshaller::looseMarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 618).

```

6.480.3.4 virtual void ac-
tivismq::wireformat::openwire::marshal::v4::MessagePullMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]

```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 619).

```

6.480.3.5 virtual int ac-
tivismq::wireformat::openwire::marshal::v4::MessagePullMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, utils::BooleanStream * bs ) throw (
decaf::io::IOException ) [virtual]

```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 620).

```
6.480.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::MessagePullMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 621).

```
6.480.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::MessagePullMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 622).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/MessagePullMarshaller.h`

6.481 activemq::transport::mock::MockTransport Class Reference

The **MockTransport** (p. 2227) defines a base level **Transport** (p. 3066) class that is intended to be used in place of an a regular protocol **Transport** (p. 3066) such as TCP.

```
#include <src/main/activemq/transport/mock/MockTransport.h>
```

Inheritance diagram for **activemq::transport::mock::MockTransport**:

Public Member Functions

- **MockTransport** (const **Pointer**< **wireformat::WireFormat** > &wireFormat, const **Pointer**< **ResponseBuilder** > &responseBuilder)
- virtual **~MockTransport** ()
- void **setResponseBuilder** (const **Pointer**< **ResponseBuilder** > &responseBuilder)
*Sets the **ResponseBuilder** (p. 2614) that this class uses to create Responses to Commands sent.*
- virtual void **oneway** (const **Pointer**< **Command** > &command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)
Sends a one-way command.
- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)
Sends the given command to the broker and then waits for the response.
- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command, unsigned int timeout) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)
Sends the given command to the broker and then waits for the response.
- virtual void **setOutgoingListener** (**TransportListener** *listener)
Sets a Listener that gets notified for every command that would have been sent by this transport to the Broker, this allows a client to verify that its messages are making it to the wire.
- virtual void **setWireFormat** (const **Pointer**< **wireformat::WireFormat** > &wireFormat **AMQCPP_UNUSED**)
Sets the WireFormat instance to use.

- **Pointer< wireformat::WireFormat > getWireFormat () const**
Gets the currently set WireFormat.
- virtual void **setTransportListener (TransportListener *listener)**
Sets the observer of asynchronous exceptions from this transport.
- virtual **TransportListener * getTransportListener () const**
Gets the observer of asynchronous exceptions from this transport.
- virtual void **fireCommand (const Pointer< Command > &command)**
Fires a Command back through this transport to its registered CommandListener if there is one.
- virtual void **fireException (const exceptions::ActiveMQException &ex)**
Fires a Exception back through this transport to its registered ExceptionListener if there is one.
- virtual void **start () throw (decaf::io::IOException)**
*Starts the **Transport** (p. 3066), the send methods of a **Transport** (p. 3066) will throw an exception if used before the **Transport** (p. 3066) is started.*
- virtual void **stop () throw (decaf::io::IOException)**
*Stops the **Transport** (p. 3066).*
- virtual void **close () throw (decaf::io::IOException)**
Closes this object and deallocates the appropriate resources.
- virtual **Transport * narrow (const std::type_info &typeId)**
*Narrows down a Chain of Transports to a specific **Transport** (p. 3066) to allow a higher level transport to skip intermediate Transports in certain circumstances.*
- virtual bool **isFaultTolerant () const**
*Is this **Transport** (p. 3066) fault tolerant, meaning that it will reconnect to a broker on disconnect.*
- virtual bool **isConnected () const**
*Is the **Transport** (p. 3066) Connected to its Broker.*
- virtual bool **isClosed () const**
*Has the **Transport** (p. 3066) been shutdown and no longer usable.*
- virtual std::string **getRemoteAddress () const**
- virtual void **reconnect (const decaf::net::URI &uri AMQCPP_UNUSED) throw (decaf::io::IOException)**
reconnect to another location
- bool **isFailOnSendMessage () const**
- void **setFailOnSendMessage (bool value)**
- int **getNumSentMessageBeforeFail () const**
- void **setNumSentMessageBeforeFail (int value)**
- int **getNumSentMessages () const**

- void **setNumSentMessages** (int value)
- bool **isFailOnReceiveMessage** () const
- void **setFailOnReceiveMessage** (bool value)
- int **getNumReceivedMessageBeforeFail** () const
- void **setNumReceivedMessageBeforeFail** (int value)
- int **getNumReceivedMessages** () const
- void **setNumReceivedMessages** (int value)
- bool **isFailOnKeepAliveSends** () const
- void **setFailOnKeepAliveSends** (bool value)
- int **getNumSentKeepAlivesBeforeFail** () const
- void **setNumSentKeepAlivesBeforeFail** (int value)
- int **getNumSentKeepAlives** () const
- void **setNumSentKeepAlives** (int value)
- bool **isFailOnStart** () const
- void **setFailOnStart** (bool value)
- bool **isFailOnStop** () const
- void **setFailOnStop** (bool value)
- bool **isFailOnClose** () const
- void **setFailOnClose** (bool value)

Static Public Member Functions

- static **MockTransport * getInstance** ()

6.481.1 Detailed Description

The **MockTransport** (p.2227) defines a base level **Transport** (p.3066) class that is intended to be used in place of an a regular protocol **Transport** (p.3066) such as TCP. This **Transport** (p.3066) assumes that it is the base **Transport** (p.3066) in the Transports stack, and destroys any Transports that are passed to it in its constructor.

This **Transport** (p.3066) defines an Interface **ResponseBuilder** (p.2614) which must be implemented by any protocol for which the **Transport** (p.3066) is used to Emulate. The **Transport** (p.3066) hands off all outbound commands to the **ResponseBuilder** (p.2614) for processing, it is up to the builder to create appropriate responses and schedule any asynchronous messages that might result from a message sent to the Broker.

6.481.2 Constructor & Destructor Documentation

- 6.481.2.1 **activemq::transport::mock::MockTransport::MockTransport** (const **Pointer**< **wireformat::WireFormat** > & *wireFormat*, const **Pointer**< **ResponseBuilder** > & *responseBuilder*)
- 6.481.2.2 **virtual activemq::transport::mock::MockTransport::~MockTransport** () [inline, virtual]

6.481.3 Member Function Documentation

- 6.481.3.1 **virtual void activemq::transport::mock::MockTransport::close** () throw (**decaf::io::IOException**) [virtual]

Closes this object and deallocates the appropriate resources.

The object is generally no longer usable after calling close.

Exceptions

IOException if an error occurs while closing.

Implements **decaf::io::Closeable** (p. 951).

6.481.3.2 `virtual void activemq::transport::mock::MockTransport::fireCommand (const Pointer< Command > & command)` [inline, virtual]

Fires a Command back through this transport to its registered CommandListener if there is one.

Parameters

command - Command to send to the Listener.

6.481.3.3 `virtual void activemq::transport::mock::MockTransport::fireException (const exceptions::ActiveMQException & ex)` [inline, virtual]

Fires a Exception back through this transport to its registered ExceptionListener if there is one.

Parameters

ex The Exception that will be passed on the the **Transport** (p. 3066) listener.

- 6.481.3.4** `static MockTransport* activemq::transport::mock::MockTransport::getInstance () [inline, static]`
- 6.481.3.5** `int activemq::transport::mock::MockTransport::getNumReceivedMessageBeforeFail () const [inline]`
- 6.481.3.6** `int activemq::transport::mock::MockTransport::getNumReceivedMessages () const [inline]`
- 6.481.3.7** `int activemq::transport::mock::MockTransport::getNumSentKeepAlives () const [inline]`
- 6.481.3.8** `int activemq::transport::mock::MockTransport::getNumSentKeepAlivesBeforeFail () const [inline]`
- 6.481.3.9** `int activemq::transport::mock::MockTransport::getNumSentMessageBeforeFail () const [inline]`
- 6.481.3.10** `int activemq::transport::mock::MockTransport::getNumSentMessages () const [inline]`
- 6.481.3.11** `virtual std::string activemq::transport::mock::MockTransport::getRemoteAddress () const [inline, virtual]`

Returns

the remote address for this connection

Implements `activemq::transport::Transport` (p. 3067).

- 6.481.3.12** `virtual TransportListener* activemq::transport::mock::MockTransport::getTransportListener () const [inline, virtual]`

Gets the observer of asynchronous exceptions from this transport.

Returns

The listener of transport events.

Implements `activemq::transport::Transport` (p. 3068).

- 6.481.3.13** `Pointer<wireformat::WireFormat> activemq::transport::mock::MockTransport::getWireFormat () const [inline]`

Gets the currently set WireFormat.

Returns

the current WireFormat object.

6.481.3.14 `virtual bool activemq::transport::mock::MockTransport::isClosed ()
const [inline, virtual]`

Has the **Transport** (p. 3066) been shutdown and no longer usable.

Returns

true if the **Transport** (p. 3066)

Implements **activemq::transport::Transport** (p. 3068).

6.481.3.15 `virtual bool activemq::transport::mock::MockTransport::isConnected ()
const [inline, virtual]`

Is the **Transport** (p. 3066) Connected to its Broker.

Returns

true if a connection has been made.

Implements **activemq::transport::Transport** (p. 3068).

6.481.3.16 `bool activemq::transport::mock::MockTransport::isFailOnClose ()
const [inline]`

6.481.3.17 `bool activemq::transport::mock::MockTransport::isFailOnKeepAliveSends ()
const [inline]`

6.481.3.18 `bool activemq::transport::mock::MockTransport::isFailOnReceiveMessage ()
const [inline]`

6.481.3.19 `bool activemq::transport::mock::MockTransport::isFailOnSendMessage ()
const [inline]`

6.481.3.20 `bool activemq::transport::mock::MockTransport::isFailOnStart ()
const [inline]`

6.481.3.21 `bool activemq::transport::mock::MockTransport::isFailOnStop ()
const [inline]`

6.481.3.22 `virtual bool activemq::transport::mock::MockTransport::isFaultTolerant ()
const [inline, virtual]`

Is this **Transport** (p. 3066) fault tolerant, meaning that it will reconnect to a broker on disconnect.

Returns

true if the **Transport** (p. 3066) is fault tolerant.

Implements **activemq::transport::Transport** (p. 3068).

6.481.3.23 `virtual Transport* activemq::transport::mock::MockTransport::narrow (const std::type_info & typeId) [inline, virtual]`

Narrows down a Chain of Transports to a specific **Transport** (p. 3066) to allow a higher level transport to skip intermediate Transports in certain circumstances.

Parameters

typeId - The type_info of the Object we are searching for.

Returns

the requested Object. or NULL if its not in this chain.

Implements **activemq::transport::Transport** (p. 3068).

6.481.3.24 `virtual void activemq::transport::mock::MockTransport::oneway (const Pointer< Command > & command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Sends a one-way command.

Does not wait for any response from the broker.

Parameters

command the command to be sent.

Exceptions

IOException if an exception occurs during writing of the command.

UnsupportedOperationException if this method is not implemented by this transport.

Implements **activemq::transport::Transport** (p. 3069).

6.481.3.25 `virtual void activemq::transport::mock::MockTransport::reconnect (const decaf::net::URI &uri AMQCPP_UNUSED) throw (decaf::io::IOException) [inline, virtual]`

reconnect to another location

Parameters

uri

Exceptions

IOException on failure of if not supported

6.481.3.26 `virtual Pointer<Response> activemq::transport::mock::MockTransport::request (const Pointer< Command > & command, unsigned int timeout) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Sends the given command to the broker and then waits for the response.

Parameters

command - The command to be sent.

timeout - The time to wait for this response.

Returns

the response from the broker.

Exceptions

IOException if an exception occurs during the read of the command.

UnsupportedOperationException if this method is not implemented by this transport.

Implements `activemq::transport::Transport` (p. 3070).

6.481.3.27 `virtual Pointer<Response> activemq::transport::mock::MockTransport::request (const Pointer< Command > & command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Sends the given command to the broker and then waits for the response.

Parameters

command the command to be sent.

Returns

the response from the broker.

Exceptions

IOException if an exception occurs during the read of the command.

UnsupportedOperationException if this method is not implemented by this transport.

Implements `activemq::transport::Transport` (p. 3069).

- 6.481.3.28 `void activemq::transport::mock::MockTransport::setFailOnClose (bool value) [inline]`
- 6.481.3.29 `void activemq::transport::mock::MockTransport::setFailOnKeepAliveSends (bool value) [inline]`
- 6.481.3.30 `void activemq::transport::mock::MockTransport::setFailOnReceiveMessage (bool value) [inline]`
- 6.481.3.31 `void activemq::transport::mock::MockTransport::setFailOnSendMessage (bool value) [inline]`
- 6.481.3.32 `void activemq::transport::mock::MockTransport::setFailOnStart (bool value) [inline]`
- 6.481.3.33 `void activemq::transport::mock::MockTransport::setFailOnStop (bool value) [inline]`
- 6.481.3.34 `void activemq::transport::mock::MockTransport::setNumReceivedMessageBeforeFail (int value) [inline]`
- 6.481.3.35 `void activemq::transport::mock::MockTransport::setNumReceivedMessages (int value) [inline]`
- 6.481.3.36 `void activemq::transport::mock::MockTransport::setNumSentKeepAlives (int value) [inline]`
- 6.481.3.37 `void activemq::transport::mock::MockTransport::setNumSentKeepAlivesBeforeFail (int value) [inline]`
- 6.481.3.38 `void activemq::transport::mock::MockTransport::setNumSentMessageBeforeFail (int value) [inline]`
- 6.481.3.39 `void activemq::transport::mock::MockTransport::setNumSentMessages (int value) [inline]`
- 6.481.3.40 `virtual void activemq::transport::mock::MockTransport::setOutgoingListener (TransportListener * listener) [inline, virtual]`

Sets a Listener that gets notified for every command that would have been sent by this transport to the Broker, this allows a client to verify that its messages are making it to the wire.

Parameters

listener - The CommandListener to notify for each message

6.481.3.41 `void activemq::transport::mock::MockTransport::setResponseBuilder (const Pointer< ResponseBuilder > & responseBuilder) [inline]`

Sets the **ResponseBuilder** (p. 2614) that this class uses to create Responses to Commands sent. These are either real Response Objects, or Commands that would have been sent Asynchronously be the Broker.

Parameters

responseBuilder - The **ResponseBuilder** (p. 2614) to use from now on.

6.481.3.42 `virtual void activemq::transport::mock::MockTransport::setTransportListener (TransportListener * listener) [inline, virtual]`

Sets the observer of asynchronous exceptions from this transport.

Parameters

listener the listener of transport events.

Implements **activemq::transport::Transport** (p. 3070).

6.481.3.43 `virtual void activemq::transport::mock::MockTransport::setWireFormat (const Pointer< wireformat::WireFormat > &wireFormat AMQCPP_UNUSED) [inline, virtual]`

Sets the WireFormat instance to use.

Parameters

wireFormat WireFormat the object used to encode / decode commands.

6.481.3.44 `virtual void activemq::transport::mock::MockTransport::start () throw (decaf::io::IOException) [virtual]`

Starts the **Transport** (p. 3066), the send methods of a **Transport** (p. 3066) will throw an exception if used before the **Transport** (p. 3066) is started.

Exceptions

IOException if and error occurs while starting the **Transport** (p. 3066).

Implements **activemq::transport::Transport** (p. 3071).

6.481.3.45 `virtual void activemq::transport::mock::MockTransport::stop () throw (decaf::io::IOException) [virtual]`

Stops the **Transport** (p. 3066).

Exceptions

IOException if an error occurs while stopping the transport.

Implements **activemq::transport::Transport** (p. 3071).

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/mock/MockTransport.h`

6.482 **activemq::transport::mock::MockTransportFactory** Class Reference

Manufactures MockTransports, which are objects that read from input streams and write to output streams.

```
#include <src/main/activemq/transport/mock/MockTransportFactory.h>
```

Inheritance diagram for **activemq::transport::mock::MockTransportFactory**:

Public Member Functions

- virtual **~MockTransportFactory** ()
- virtual **Pointer< Transport > create** (const **decaf::net::URI** &location) throw (exceptions::ActiveMQException)
*Creates a fully configured **Transport** (p. 3066) instance which could be a chain of filters and transports.*
- virtual **Pointer< Transport > createComposite** (const **decaf::net::URI** &location) throw (exceptions::ActiveMQException)
*Creates a slimed down **Transport** (p. 3066) instance which can be used in composite transport instances.*

Protected Member Functions

- virtual **Pointer< Transport > doCreateComposite** (const **decaf::net::URI** &location, const **Pointer< wireformat::WireFormat >** &wireFormat, const **decaf::util::Properties** &properties) throw (exceptions::ActiveMQException)
*Creates a slimed down **Transport** (p. 3066) instance which can be used in composite transport instances.*

6.482.1 Detailed Description

Manufactures MockTransports, which are objects that read from input streams and write to output streams.

6.482.2 Constructor & Destructor Documentation

6.482.2.1 virtual
activemq::transport::mock::MockTransportFactory::~MockTransportFactory
() [inline, virtual]

6.482.3 Member Function Documentation

6.482.3.1 virtual Pointer<Transport> ac-
tivemq::transport::mock::MockTransportFactory::create
(const decaf::net::URI & *location*) throw (
exceptions::ActiveMQException) [virtual]

Creates a fully configured **Transport** (p.3066) instance which could be a chain of filters and transports.

Parameters

location - URI location to connect to plus any properties to assign.

Exceptions

ActiveMQException if an error occurs

Implements **activemq::transport::TransportFactory** (p.3072).

6.482.3.2 virtual Pointer<Transport> ac-
tivemq::transport::mock::MockTransportFactory::createComposite
(const decaf::net::URI & *location*) throw (
exceptions::ActiveMQException) [virtual]

Creates a slimed down **Transport** (p.3066) instance which can be used in composite transport instances.

Parameters

location - URI location to connect to plus any properties to assign.

Exceptions

ActiveMQException if an error occurs

Implements **activemq::transport::TransportFactory** (p.3072).

6.482.3.3 virtual Pointer<Transport> ac-
tivemq::transport::mock::MockTransportFactory::doCreateComposite
(const decaf::net::URI & *location*, const Pointer<
wireformat::WireFormat > & *wireFormat*, const decaf::util::Properties
& *properties*) throw (exceptions::ActiveMQException) [protected,
virtual]

Creates a slimed down **Transport** (p.3066) instance which can be used in composite transport instances.

Parameters

location - URI location to connect to.

wireFormat - the assigned WireFormat for the new **Transport** (p. 3066).

properties - Properties to apply to the transport.

Returns

Pointer to a new **Transport** (p. 3066) instance.

Exceptions

ActiveMQException if an error occurs

The documentation for this class was generated from the following file:

- src/main/activemq/transport/mock/**MockTransportFactory.h**

6.483 decaf::util::concurrent::Mutex Class Reference

Mutex (p. 2239) object that offers recursive support on all platforms as well as providing the ability to use the standard wait / notify pattern used in languages like Java.

#include <src/main/decaf/util/concurrent/Mutex.h>

Inheritance diagram for decaf::util::concurrent::Mutex:

Public Member Functions

- **Mutex** ()
- virtual ~**Mutex** ()
- virtual void **lock** () throw (decaf::lang::exceptions::RuntimeException)
Locks the object.
- virtual bool **tryLock** () throw (decaf::lang::exceptions::RuntimeException)
*Attempts to **Lock** (p. 1903) the object, if the lock is already held by another thread than this method returns false.*
- virtual void **unlock** () throw (decaf::lang::exceptions::RuntimeException)
Unlocks the object.
- virtual void **wait** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (long long millisecs) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.

- virtual void **wait** (long long millisecs, int nanos) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)

Waits on a signal from this object, which is generated by a call to Notify.

- virtual void **notify** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)

Signals a waiter on this object that it can now wake up and continue.

- virtual void **notifyAll** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)

Signals the waiters on this object that it can now wake up and continue.

6.483.1 Detailed Description

Mutex (p.2239) object that offers recursive support on all platforms as well as providing the ability to use the standard wait / notify pattern used in languages like Java.

Since

1.0

6.483.2 Constructor & Destructor Documentation

6.483.2.1 decaf::util::concurrent::Mutex::Mutex ()

6.483.2.2 virtual decaf::util::concurrent::Mutex::~~Mutex () [virtual]

6.483.3 Member Function Documentation

6.483.3.1 virtual void decaf::util::concurrent::Mutex::lock () throw (decaf::lang::exceptions::RuntimeException) [virtual]

Locks the object.

Exceptions

RuntimeException if an error occurs while locking the object.

Implements decaf::util::concurrent::Synchronizable (p.2932).

Referenced by decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >::lock(), decaf::util::StlMap< std::string, cms::Topic * >::lock(), decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::lock(), and decaf::util::AbstractCollection< Pointer< BackupTransport > >::lock().

6.483.3.2 `virtual void decaf::util::concurrent::Mutex::notify ()
 throw (decaf::lang::exceptions::RuntimeException,
 decaf::lang::exceptions::IllegalMonitorStateException) [virtual]`

Signals a waiter on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

IllegalMonitorStateException - if the current thread is not the owner of the the **Synchronizable** (p. 2930) Object.

RuntimeException if an error occurs while notifying one of the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 2933).

Referenced by `decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >::notify()`, `decaf::util::StlMap< std::string, cms::Topic * >::notify()`, `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::notify()`, and `decaf::util::AbstractCollection< Pointer< BackupTransport > >::notify()`.

6.483.3.3 `virtual void decaf::util::concurrent::Mutex::notifyAll
 () throw (decaf::lang::exceptions::RuntimeException,
 decaf::lang::exceptions::IllegalMonitorStateException) [virtual]`

Signals the waiters on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

IllegalMonitorStateException - if the current thread is not the owner of the the **Synchronizable** (p. 2930) Object.

RuntimeException if an error occurs while notifying the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 2934).

Referenced by `decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >::notifyAll()`, `decaf::util::StlMap< std::string, cms::Topic * >::notifyAll()`, `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::notifyAll()`, and `decaf::util::AbstractCollection< Pointer< BackupTransport > >::notifyAll()`.

6.483.3.4 `virtual bool decaf::util::concurrent::Mutex::tryLock () throw (decaf::lang::exceptions::RuntimeException) [virtual]`

Attempts to **Lock** (p.1903) the object, if the lock is already held by another thread than this method returns false.

Returns

true if the lock was acquired, false if it is already held by another thread.

Exceptions

RuntimeException if an error occurs while locking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2935).

Referenced by `decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >::tryLock()`, `decaf::util::StlMap< std::string, cms::Topic * >::tryLock()`, `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::tryLock()`, and `decaf::util::AbstractCollection< Pointer< BackupTransport > >::tryLock()`.

6.483.3.5 virtual void decaf::util::concurrent::Mutex::unlock () throw (decaf::lang::exceptions::RuntimeException) [virtual]

Unlocks the object.

Exceptions

RuntimeException if an error occurs while unlocking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2936).

Referenced by `decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >::unlock()`, `decaf::util::StlMap< std::string, cms::Topic * >::unlock()`, `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::unlock()`, and `decaf::util::AbstractCollection< Pointer< BackupTransport > >::unlock()`.

6.483.3.6 virtual void decaf::util::concurrent::Mutex::wait () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException) [virtual]

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling.

Exceptions

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the **Synchronizable** (p. 2930) Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2937).

Referenced by `decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >::wait()`, `decaf::util::StlMap< std::string, cms::Topic * >::wait()`, `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::wait()`, and `decaf::util::AbstractCollection< Pointer< BackupTransport > >::wait()`.

6.483.3.7 `virtual void decaf::util::concurrent::Mutex::wait (long long millisecs) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException) [virtual]`

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval.

Parameters

millisecs the time in milliseconds to wait, or WAIT_INFINITE

Exceptions

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the **Synchronizable** (p. 2930) Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2938).

6.483.3.8 `virtual void decaf::util::concurrent::Mutex::wait (long long millisecs, int nanos) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException) [virtual]`

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

Parameters

millisecs the time in milliseconds to wait, or WAIT_INFINITE

nanos additional time in nanoseconds with a range of 0-999999

Exceptions

IllegalArgumentException if an error occurs or the nanos argument is not in the range of [0-999999]

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the **Synchronizable** (p. 2930) Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2940).

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**Mutex.h**

6.484 decaf::util::concurrent::MutexHandle Class Reference

```
#include <src/main/decaf/internal/util/concurrent/unix/MutexHandle.h>
```

Public Member Functions

- **MutexHandle** ()
- **~MutexHandle** ()
- **MutexHandle** ()
- **~MutexHandle** ()

Data Fields

- pthread_mutex_t **mutex**
- volatile long long **lock_owner**
- volatile long long **lock_count**
- CRITICAL_SECTION **mutex**

6.484.1 Constructor & Destructor Documentation

6.484.1.1 decaf::util::concurrent::MutexHandle::MutexHandle () [inline]

6.484.1.2 decaf::util::concurrent::MutexHandle::~~MutexHandle () [inline]

6.484.1.3 decaf::util::concurrent::MutexHandle::MutexHandle () [inline]

6.484.1.4 decaf::util::concurrent::MutexHandle::~~MutexHandle () [inline]

6.484.2 Field Documentation

6.484.2.1 volatile long long decaf::util::concurrent::MutexHandle::lock_count

6.484.2.2 volatile long long decaf::util::concurrent::MutexHandle::lock_owner

6.484.2.3 CRITICAL_SECTION decaf::util::concurrent::MutexHandle::mutex

6.484.2.4 pthread_mutex_t decaf::util::concurrent::MutexHandle::mutex

The documentation for this class was generated from the following files:

- src/main/decaf/internal/util/concurrent/unix/**MutexHandle.h**
- src/main/decaf/internal/util/concurrent/windows/**MutexHandle.h**

6.485 decaf::internal::util::concurrent::MutexImpl Class Reference

```
#include <src/main/decaf/internal/util/concurrent/MutexImpl.h>
```

Static Public Member Functions

- static **decaf::util::concurrent::MutexHandle * create** ()
Creates a Reentrant Mutex and returns the handle, throws a Runtime Exception if the Mutex cannot be created for some reason.
- static void **destroy** (decaf::util::concurrent::MutexHandle *handle)
Destroy a previously create Mutex instance.
- static void **lock** (decaf::util::concurrent::MutexHandle *handle)
Locks the Mutex.
- static bool **trylock** (decaf::util::concurrent::MutexHandle *handle)
Tries to lock the Mutex.
- static void **unlock** (decaf::util::concurrent::MutexHandle *handle)
Unlocks the Mutex allowing other Thread to then acquire the Lock on it.

6.485.1 Member Function Documentation

6.485.1.1 static decaf::util::concurrent::MutexHandle* decaf::internal::util::concurrent::MutexImpl::create () [static]

Creates a Reentrant Mutex and returns the handle, throws a Runtime Exception if the Mutex cannot be created for some reason.

Returns

handle to a newly created Mutex.

6.485.1.2 static void decaf::internal::util::concurrent::MutexImpl::destroy (decaf::util::concurrent::MutexHandle * *handle*) [static]

Destroy a previously create Mutex instance.

Parameters

mutex The Mutex instance to be destroyed.

6.485.1.3 static void decaf::internal::util::concurrent::MutexImpl::lock (decaf::util::concurrent::MutexHandle * *handle*) [static]

Locks the Mutex.

If the Mutex is already locked by another thread this method blocks until the Mutex becomes unlocked and this thread acquires the lock.

Parameters

handle the handle to the Mutex to Lock.

6.485.1.4 `static bool decaf::internal::util::concurrent::MutexImpl::trylock (decaf::util::concurrent::MutexHandle * handle) [static]`

Tries to lock the Mutex.

If the Mutex is unlocked this Thread acquires the lock on the Mutex and this method returns true, if the Mutex is already locked then the lock is not acquired and this method returns false.

Parameters

handle the handle to the Mutex to attempt to Lock.

Returns

true if the lock was acquired false otherwise.

6.485.1.5 `static void decaf::internal::util::concurrent::MutexImpl::unlock (decaf::util::concurrent::MutexHandle * handle) [static]`

Unlocks the Mutex allowing other Thread to then acquire the Lock on it.

Parameters

handle the handle to the Mutex to attempt to Lock.

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/util/concurrent/MutexImpl.h`

6.486 activemq::commands::NetworkBridgeFilter Class Reference

```
#include <src/main/activemq/commands/NetworkBridgeFilter.h>
```

Inheritance diagram for activemq::commands::NetworkBridgeFilter:

Public Member Functions

- `NetworkBridgeFilter ()`
- `virtual ~NetworkBridgeFilter ()`
- `virtual unsigned char getDataStructureType () const`
Get the unique identifier that this object and its own Marshaler share.
- `virtual NetworkBridgeFilter * cloneDataStructure () const`
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- `virtual void copyDataStructure (const DataStructure *src)`
Copy the contents of the passed object into this object's members, overwriting any existing data.

- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1372) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent.*
- virtual int **getNetworkTTL** () const
- virtual void **setNetworkTTL** (int **networkTTL**)
- virtual const **Pointer**< **BrokerId** > & **getNetworkBrokerId** () const
- virtual **Pointer**< **BrokerId** > & **getNetworkBrokerId** ()
- virtual void **setNetworkBrokerId** (const **Pointer**< **BrokerId** > &**networkBrokerId**)

Static Public Attributes

- static const unsigned char **ID_NETWORKBRIDGEFILTER** = 91

Protected Member Functions

- **NetworkBridgeFilter** (const **NetworkBridgeFilter** &)
- **NetworkBridgeFilter** & **operator=** (const **NetworkBridgeFilter** &)

Protected Attributes

- int **networkTTL**
- **Pointer**< **BrokerId** > **networkBrokerId**

6.486.1 Constructor & Destructor Documentation

6.486.1.1 **activemq::commands::NetworkBridgeFilter::NetworkBridgeFilter** (const **NetworkBridgeFilter** &) [inline, protected]

6.486.1.2 **activemq::commands::NetworkBridgeFilter::NetworkBridgeFilter** ()

6.486.1.3 virtual
activemq::commands::NetworkBridgeFilter::~~NetworkBridgeFilter ()
[virtual]

6.486.2 Member Function Documentation

6.486.2.1 virtual **NetworkBridgeFilter*** **activemq::commands::NetworkBridgeFilter::cloneDataStructure** () const
[virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1373).

6.486.2.2 `virtual void activemq::commands::NetworkBridgeFilter::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

Implements **activemq::commands::DataStructure** (p. 1374).

6.486.2.3 `virtual bool activemq::commands::NetworkBridgeFilter::equals (const DataStructure * value) const [virtual]`

Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Implements **activemq::commands::DataStructure** (p. 1374).

6.486.2.4 `virtual unsigned char activemq::commands::NetworkBridgeFilter::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataStructure** (p. 1372) type copy.

Implements **activemq::commands::DataStructure** (p. 1375).

- 6.486.2.5 `virtual const Pointer<BrokerId>& activemq::commands::NetworkBridgeFilter::getNetworkBrokerId () const [virtual]`
- 6.486.2.6 `virtual Pointer<BrokerId>& activemq::commands::NetworkBridgeFilter::getNetworkBrokerId () [virtual]`
- 6.486.2.7 `virtual int activemq::commands::NetworkBridgeFilter::getNetworkTTL () const [virtual]`
- 6.486.2.8 `NetworkBridgeFilter& activemq::commands::NetworkBridgeFilter::operator= (const NetworkBridgeFilter &) [inline, protected]`
- 6.486.2.9 `virtual void activemq::commands::NetworkBridgeFilter::setNetworkBrokerId (const Pointer< BrokerId > & networkBrokerId) [virtual]`
- 6.486.2.10 `virtual void activemq::commands::NetworkBridgeFilter::setNetworkTTL (int networkTTL) [virtual]`
- 6.486.2.11 `virtual std::string activemq::commands::NetworkBridgeFilter::toString () const [virtual]`

Returns a string containing the information for this **DataStructure** (p.1372) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from `activemq::commands::BaseDataStructure` (p.662).

6.486.3 Field Documentation

- 6.486.3.1 `const unsigned char activemq::commands::NetworkBridgeFilter::ID_ - NETWORKBRIDGEFILTER = 91 [static]`
- 6.486.3.2 `Pointer<BrokerId> activemq::commands::NetworkBridgeFilter::networkBrokerId [protected]`
- 6.486.3.3 `int activemq::commands::NetworkBridgeFilter::networkTTL [protected]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/NetworkBridgeFilter.h`

6.487 activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller Class Reference

Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2250).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/NetworkBridgeFilterMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller:

Public Member Functions

- **NetworkBridgeFilterMarshaller** ()
- virtual **~NetworkBridgeFilterMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.487.1 Detailed Description

Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p.2250).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.487.2 Constructor & Destructor Documentation

6.487.2.1 `activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller::NetworkBridgeFilterMarshaller()` [inline]

6.487.2.2 `virtual activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller::~~NetworkBridgeFilterMarshaller()` [inline, virtual]

6.487.3 Member Function Documentation

6.487.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p.1331).

6.487.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p.1336).

6.487.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut)` throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1342).

6.487.3.4 virtual void **activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller::looseUnmarshal**
(**OpenWireFormat** * *wireFormat*, **commands::DataStructure** *
dataStructure, **decaf::io::DataInputStream** * *dataIn*) throw (
decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1348).

6.487.3.5 virtual int **activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller::tightMarshal**
(**OpenWireFormat** * *wireFormat*, **commands::DataStructure**
* *dataStructure*, **utils::BooleanStream** * *bs*) throw (
decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1354).

6.487.3.6 `virtual void activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller::tightMarshal2`
`(OpenWireFormat * wireFormat, commands::DataStructure`
`* dataStructure, decaf::io::DataOutputStream * dataOut,`
`utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1360).

6.487.3.7 `virtual void activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller::tightUnmarshal`
`(OpenWireFormat * wireFormat, commands::DataStructure`
`* dataStructure, decaf::io::DataInputStream * dataIn,`
`utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1366).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/NetworkBridgeFilterMarshaller.h`

6.488 `activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilter` Class Reference

Marshaling code for Open Wire Format for `NetworkBridgeFilterMarshaller` (p. 2253).

#include <src/main/activemq/wireformat/openwire/marshal/v3/NetworkBridgeFilterMarshaller.h>

Inheritance diagram for activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller:

Public Member Functions

- **NetworkBridgeFilterMarshaller** ()
- virtual **~NetworkBridgeFilterMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.488.1 Detailed Description

Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2253).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.488.2 Constructor & Destructor Documentation

6.488.2.1 `activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller::NetworkBridgeFilterMarshaller () [inline]`

6.488.2.2 `virtual activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller::~~NetworkBridgeFilterMarshaller () [inline, virtual]`

6.488.3 Member Function Documentation

6.488.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.488.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.488.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1342).

6.488.3.4 virtual void activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1348).

6.488.3.5 virtual int activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller::tightMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1354).

6.488.3.6 virtual void activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller::tightMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1360).

```
6.488.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller::tightUnmarsh
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1366).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/NetworkBridgeFilterMarshaller.h`

6.489 **activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller** Class Reference

Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2257).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/NetworkBridgeFilterMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller**:

Public Member Functions

- **NetworkBridgeFilterMarshaller** ()
- virtual **~NetworkBridgeFilterMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.489.1 Detailed Description

Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2257).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.489.2 Constructor & Destructor Documentation

6.489.2.1 `activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller::NetworkBridgeFilterMarshaller () [inline]`

6.489.2.2 `virtual activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller::~~NetworkBridgeFilterMarshaller () [inline, virtual]`

6.489.3 Member Function Documentation

6.489.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.489.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.489.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1342).

6.489.3.4 virtual void activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1348).

6.489.3.5 virtual int activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller::tightMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1354).

6.489.3.6 virtual void activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller::tightMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1360).

```
6.489.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller::tightUnmarsh
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1366).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/NetworkBridgeFilterMarshaller.h`

6.490 **activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller** Class Reference

Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2261).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/NetworkBridgeFilterMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller**:

Public Member Functions

- **NetworkBridgeFilterMarshaller** ()
- virtual **~NetworkBridgeFilterMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.490.1 Detailed Description

Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p.2261).
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.490.2 Constructor & Destructor Documentation

6.490.2.1 `activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller::NetworkBridgeFilterMarshaller () [inline]`

6.490.2.2 `virtual activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller::~~NetworkBridgeFilterMarshaller () [inline, virtual]`

6.490.3 Member Function Documentation

6.490.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.490.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.490.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1342).

6.490.3.4 virtual void activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1348).

6.490.3.5 virtual int activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller::tightMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1354).

6.490.3.6 virtual void activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller::tightMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1360).

```
6.490.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller::tightUnmarsh
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1366).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/NetworkBridgeFilterMarshaller.h

6.491 activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller Class Reference

Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2265).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/NetworkBridgeFilterMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller**:

Public Member Functions

- **NetworkBridgeFilterMarshaller** ()
- virtual **~NetworkBridgeFilterMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.491.1 Detailed Description

Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p.2265).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.491.2 Constructor & Destructor Documentation

6.491.2.1 `activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller::NetworkBridgeFilterMarshaller () [inline]`

6.491.2.2 `virtual activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller::~~NetworkBridgeFilterMarshaller () [inline, virtual]`

6.491.3 Member Function Documentation

6.491.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.491.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.491.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1342).

6.491.3.4 virtual void activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1348).

6.491.3.5 virtual int activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller::tightMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1354).

6.491.3.6 virtual void activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller::tightMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1360).

6.491.3.7 `virtual void activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1366).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/NetworkBridgeFilterMarshaller.h`

6.492 decaf::net::NoRouteToHostException Class Reference

```
#include <src/main/decaf/net/NoRouteToHostException.h>
```

Inheritance diagram for `decaf::net::NoRouteToHostException`:

Public Member Functions

- **NoRouteToHostException** () throw ()
Default Constructor.

- **NoRouteToHostException** (const Exception &ex) throw ()
Conversion Constructor from some other Exception.
- **NoRouteToHostException** (const NoRouteToHostException &ex) throw ()
Copy Constructor.
- **NoRouteToHostException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **NoRouteToHostException** (const std::exception *cause) throw ()
Constructor.
- **NoRouteToHostException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **NoRouteToHostException * clone** () const
Clones this exception.
- virtual ~**NoRouteToHostException** () throw ()

6.492.1 Constructor & Destructor Documentation

6.492.1.1 decaf::net::NoRouteToHostException::NoRouteToHostException () throw () [inline]

Default Constructor.

6.492.1.2 decaf::net::NoRouteToHostException::NoRouteToHostException (const Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

Parameters

ex An exception that should become this type of Exception

6.492.1.3 decaf::net::NoRouteToHostException::NoRouteToHostException (const NoRouteToHostException & ex) throw () [inline]

Copy Constructor.

Parameters

ex An exception that should become this type of Exception

6.492.1.4 `decaf::net::NoRouteToHostException::NoRouteToHostException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.492.1.5 `decaf::net::NoRouteToHostException::NoRouteToHostException (const std::exception * cause) throw () [inline]`

Constructor.

Parameters

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.492.1.6 `decaf::net::NoRouteToHostException::NoRouteToHostException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.492.1.7 `virtual decaf::net::NoRouteToHostException::~~NoRouteToHostException () throw () [inline, virtual]`

6.492.2 Member Function Documentation

6.492.2.1 `virtual NoRouteToHostException* decaf::net::NoRouteToHostException::clone () const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::net::SocketException** (p. 2794).

The documentation for this class was generated from the following file:

- src/main/decaf/net/**NoRouteToHostException.h**

6.493 decaf::security::NoSuchAlgorithmException Class Reference

```
#include <src/main/decaf/security/NoSuchAlgorithmException.h>
```

Inheritance diagram for decaf::security::NoSuchAlgorithmException:

Public Member Functions

- **NoSuchAlgorithmException** () throw ()
Default Constructor.
- **NoSuchAlgorithmException** (const Exception &ex) throw ()
Conversion Constructor from some other Exception.
- **NoSuchAlgorithmException** (const NoSuchAlgorithmException &ex) throw ()
Copy Constructor.
- **NoSuchAlgorithmException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **NoSuchAlgorithmException** (const std::exception *cause) throw ()
Constructor.
- **NoSuchAlgorithmException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **NoSuchAlgorithmException * clone** () const
Clones this exception.
- virtual **~NoSuchAlgorithmException** () throw ()

6.493.1 Constructor & Destructor Documentation

6.493.1.1 `decaf::security::NoSuchAlgorithmException::NoSuchAlgorithmException () throw () [inline]`

Default Constructor.

6.493.1.2 `decaf::security::NoSuchAlgorithmException::NoSuchAlgorithmException (const Exception & ex) throw () [inline]`

Conversion Constructor from some other Exception.

Parameters

ex An exception that should become this type of Exception

6.493.1.3 `decaf::security::NoSuchAlgorithmException::NoSuchAlgorithmException (const NoSuchAlgorithmException & ex) throw () [inline]`

Copy Constructor.

Parameters

ex An exception that should become this type of Exception

6.493.1.4 `decaf::security::NoSuchAlgorithmException::NoSuchAlgorithmException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.493.1.5 `decaf::security::NoSuchAlgorithmException::NoSuchAlgorithmException (const std::exception * cause) throw () [inline]`

Constructor.

Parameters

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.493.1.6 `decaf::security::NoSuchAlgorithmException::NoSuchAlgorithmException`
 (`const char * file`, `const int lineNumber`, `const char * msg`, ...)
 throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file name where exception occurs

lineNumber line number where the exception occurred.

msg message to report

... list of primitives that are formatted into the message

6.493.1.7 `virtual`
`decaf::security::NoSuchAlgorithmException::~~NoSuchAlgorithmException`
 () throw () [inline, virtual]

6.493.2 Member Function Documentation

6.493.2.1 `virtual NoSuchAlgorithmException* de-`
`caf::security::NoSuchAlgorithmException::clone ()`
`const` [inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

A deep copy of this exception.

Reimplemented from `decaf::security::GeneralSecurityException` (p. 1604).

The documentation for this class was generated from the following file:

- `src/main/decaf/security/NoSuchAlgorithmException.h`

6.494 decaf::lang::exceptions::NoSuchElementException Class Reference

```
#include <src/main/decaf/lang/exceptions/NoSuchElementException.h>
```

Inheritance diagram for `decaf::lang::exceptions::NoSuchElementException`:

Public Member Functions

- **NoSuchElementException** () throw ()
Default Constructor.
- **NoSuchElementException** (const **Exception** &ex) throw ()
*Conversion Constructor from some other **Exception** (p. 1476).*
- **NoSuchElementException** (const **NoSuchElementException** &ex) throw ()
Copy Constructor.
- **NoSuchElementException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **NoSuchElementException** (const std::exception *cause) throw ()
Constructor.
- **NoSuchElementException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **NoSuchElementException** * clone () const
Clones this exception.
- virtual ~**NoSuchElementException** () throw ()

6.494.1 Constructor & Destructor Documentation

6.494.1.1 **decaf::lang::exceptions::NoSuchElementException::NoSuchElementException**
() throw () [inline]

Default Constructor.

6.494.1.2 **decaf::lang::exceptions::NoSuchElementException::NoSuchElementException**
(const **Exception** & ex) throw () [inline]

Conversion Constructor from some other **Exception** (p. 1476).

Parameters

ex The **Exception** (p. 1476) whose data is to be copied into this one.

6.494.1.3 **decaf::lang::exceptions::NoSuchElementException::NoSuchElementException**
(const **NoSuchElementException** & ex) throw () [inline]

Copy Constructor.

Parameters

ex The **Exception** (p. 1476) whose data is to be copied into this one.

6.494.1.4 `decaf::lang::exceptions::NoSuchElementException::NoSuchElementException`
(`const char * file`, `const int lineNumber`, `const std::exception * cause`, `const char * msg`, ...) `throw ()` [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.494.1.5 `decaf::lang::exceptions::NoSuchElementException::NoSuchElementException`
(`const std::exception * cause`) `throw ()` [inline]

Constructor.

Parameters

cause Pointer (p.2343) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.494.1.6 `decaf::lang::exceptions::NoSuchElementException::NoSuchElementException`
(`const char * file`, `const int lineNumber`, `const char * msg`, ...)
`throw ()` [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.494.1.7 `virtual`
`decaf::lang::exceptions::NoSuchElementException::~~NoSuchElementException`
() `throw ()` [inline, virtual]

6.494.2 Member Function Documentation

6.494.2.1 `virtual NoSuchElementException* decaf::lang::exceptions::NoSuchElementException::clone` (
) `const` [inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

an new **Exception** (p. 1476) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1479).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/exceptions/NoSuchElementException.h`

6.495 decaf::security::NoSuchProviderException Class Reference

```
#include <src/main/decaf/security/NoSuchProviderException.h>
```

Inheritance diagram for `decaf::security::NoSuchProviderException`:

Public Member Functions

- **NoSuchProviderException** () throw ()
Default Constructor.
- **NoSuchProviderException** (const Exception &ex) throw ()
Conversion Constructor from some other Exception.
- **NoSuchProviderException** (const NoSuchProviderException &ex) throw ()
Copy Constructor.
- **NoSuchProviderException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **NoSuchProviderException** (const std::exception *cause) throw ()
Constructor.
- **NoSuchProviderException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **NoSuchProviderException** * clone () const
Clones this exception.
- virtual ~**NoSuchProviderException** () throw ()

6.495.1 Constructor & Destructor Documentation

6.495.1.1 decaf::security::NoSuchProviderException::NoSuchProviderException () throw () [inline]

Default Constructor.

6.495.1.2 decaf::security::NoSuchProviderException::NoSuchProviderException (const Exception & *ex*) throw () [inline]

Conversion Constructor from some other Exception.

Parameters

ex An exception that should become this type of Exception

6.495.1.3 decaf::security::NoSuchProviderException::NoSuchProviderException (const NoSuchProviderException & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex An exception that should become this type of Exception

6.495.1.4 decaf::security::NoSuchProviderException::NoSuchProviderException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.495.1.5 decaf::security::NoSuchProviderException::NoSuchProviderException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.495.1.6 `decaf::security::NoSuchProviderException::NoSuchProviderException (const char * file, const int lineNumber, const char * msg, ...) throw ()` [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file name where exception occurs

lineNumber line number where the exception occurred.

msg message to report

... list of primitives that are formatted into the message

6.495.1.7 `virtual decaf::security::NoSuchProviderException::~~NoSuchProviderException () throw ()` [inline, virtual]

6.495.2 Member Function Documentation

6.495.2.1 `virtual NoSuchProviderException* decaf::security::NoSuchProviderException::clone () const` [inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

A deep copy of this exception.

Reimplemented from `decaf::security::GeneralSecurityException` (p. 1604).

The documentation for this class was generated from the following file:

- `src/main/decaf/security/NoSuchProviderException.h`

6.496 decaf::lang::exceptions::NullPointerException Class Reference

```
#include <src/main/decaf/lang/exceptions/NullPointerException.h>
```

Inheritance diagram for `decaf::lang::exceptions::NullPointerException`:

Public Member Functions

- **NullPointerException** () throw ()
Default Constructor.
- **NullPointerException** (const **Exception** &ex) throw ()
*Conversion Constructor from some other **Exception** (p. 1476).*
- **NullPointerException** (const **NullPointerException** &ex) throw ()
Copy Constructor.
- **NullPointerException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **NullPointerException** (const std::exception *cause) throw ()
Constructor.
- **NullPointerException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **NullPointerException** * **clone** () const
Clones this exception.
- virtual ~**NullPointerException** () throw ()

6.496.1 Constructor & Destructor Documentation

6.496.1.1 decaf::lang::exceptions::NullPointerException::NullPointerException () throw () [inline]

Default Constructor.

6.496.1.2 decaf::lang::exceptions::NullPointerException::NullPointerException (const Exception & ex) throw () [inline]

Conversion Constructor from some other **Exception** (p. 1476).

Parameters

ex The **Exception** (p. 1476) whose data is to be copied into this one.

6.496.1.3 decaf::lang::exceptions::NullPointerException::NullPointerException (const NullPointerException & ex) throw () [inline]

Copy Constructor.

Parameters

ex The **Exception** (p. 1476) whose data is to be copied into this one.

6.496.1.4 `decaf::lang::exceptions::NullPointerException::NullPointerException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs
lineNumber The line number where the exception occurred.
cause The exception that was the cause for this one to be thrown.
msg The message to report
... list of primitives that are formatted into the message

6.496.1.5 `decaf::lang::exceptions::NullPointerException::NullPointerException (const std::exception * cause) throw () [inline]`

Constructor.

Parameters

cause **Pointer** (p.2343) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.496.1.6 `decaf::lang::exceptions::NullPointerException::NullPointerException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs
lineNumber The line number where the exception occurred.
msg The message to report
... list of primitives that are formatted into the message

6.496.1.7 `virtual decaf::lang::exceptions::NullPointerException::~~NullPointerException () throw () [inline, virtual]`

6.496.2 Member Function Documentation

6.496.2.1 `virtual NullPointerException* decaf::lang::exceptions::NullPointerException::clone () const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

an new **Exception** (p. 1476) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1479).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/exceptions/NullPointerException.h`

6.497 decaf::lang::Number Class Reference

The abstract class **Number** (p. 2282) is the superclass of classes **Byte** (p. 776), **Double** (p. 1441), **Float** (p. 1544), **Integer** (p. 1652), **Long** (p. 1934), and **Short** (p. 2729).

```
#include <src/main/decaf/lang/Number.h>
```

Inheritance diagram for decaf::lang::Number:

Public Member Functions

- virtual `~Number ()`
- virtual unsigned char **byteValue ()** const
Answers the byte value which the receiver represents.
- virtual double **doubleValue ()** const =0
Answers the double value which the receiver represents.
- virtual float **floatValue ()** const =0
Answers the float value which the receiver represents.
- virtual int **intValue ()** const =0
Answers the int value which the receiver represents.
- virtual long long **longValue ()** const =0
Answers the long value which the receiver represents.
- virtual short **shortValue ()** const
Answers the short value which the receiver represents.

6.497.1 Detailed Description

The abstract class **Number** (p. 2282) is the superclass of classes **Byte** (p. 776), **Double** (p. 1441), **Float** (p. 1544), **Integer** (p. 1652), **Long** (p. 1934), and **Short** (p. 2729). Subclasses of **Number** (p. 2282) must provide methods to convert the represented numeric value to byte, double, float, int, long, and short.

6.497.2 Constructor & Destructor Documentation

6.497.2.1 `virtual decaf::lang::Number::~~Number () [inline, virtual]`

6.497.3 Member Function Documentation

6.497.3.1 `virtual unsigned char decaf::lang::Number::byteValue () const [inline, virtual]`

Answers the byte value which the receiver represents.

Returns

byte the value of the receiver.

6.497.3.2 `virtual double decaf::lang::Number::doubleValue () const [pure virtual]`

Answers the double value which the receiver represents.

Returns

double the value of the receiver.

Implemented in `decaf::util::concurrent::atomic::AtomicInteger` (p. 584).

6.497.3.3 `virtual float decaf::lang::Number::floatValue () const [pure virtual]`

Answers the float value which the receiver represents.

Returns

float the value of the receiver.

Implemented in `decaf::util::concurrent::atomic::AtomicInteger` (p. 584).

6.497.3.4 `virtual int decaf::lang::Number::intValue () const [pure virtual]`

Answers the int value which the receiver represents.

Returns

int the value of the receiver.

Implemented in `decaf::util::concurrent::atomic::AtomicInteger` (p. 586).

6.497.3.5 `virtual long long decaf::lang::Number::longValue () const [pure virtual]`

Answers the long value which the receiver represents.

Returns

long long the value of the receiver.

Implemented in `decaf::util::concurrent::atomic::AtomicInteger` (p. 586).

6.497.3.6 `virtual short decaf::lang::Number::shortValue () const [inline, virtual]`

Answers the short value which the receiver represents.

Returns

short the value of the receiver.

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Number.h`

6.498 decaf::lang::exceptions::NumberFormatException Class Reference

```
#include <src/main/decaf/lang/exceptions/NumberFormatException.h>
```

Inheritance diagram for `decaf::lang::exceptions::NumberFormatException`:

Public Member Functions

- **NumberFormatException** ()
Default Constructor.
- **NumberFormatException** (const **Exception** &ex) throw ()
*Conversion Constructor from some other **Exception** (p. 1476).*
- **NumberFormatException** (const **NumberFormatException** &ex) throw ()
Copy Constructor.
- **NumberFormatException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **NumberFormatException** (const std::exception *cause) throw ()
Constructor.
- **NumberFormatException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.

- virtual **NumberFormatException** * **clone** () const
Clones this exception.
- virtual ~**NumberFormatException** () throw ()

6.498.1 Constructor & Destructor Documentation

6.498.1.1 `decaf::lang::exceptions::NumberFormatException::NumberFormatException () [inline]`

Default Constructor.

Referenced by `clone()`.

6.498.1.2 `decaf::lang::exceptions::NumberFormatException::NumberFormatException (const Exception & ex) throw () [inline]`

Conversion Constructor from some other **Exception** (p. 1476).

Parameters

ex The **Exception** (p. 1476) whose data is to be copied into this one.

6.498.1.3 `decaf::lang::exceptions::NumberFormatException::NumberFormatException (const NumberFormatException & ex) throw () [inline]`

Copy Constructor.

Parameters

ex The **Exception** (p. 1476) whose data is to be copied into this one.

6.498.1.4 `decaf::lang::exceptions::NumberFormatException::NumberFormatException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

References `decaf::lang::Exception::buildMessage()`, and `decaf::lang::Exception::setMark()`.

6.498.1.5 `decaf::lang::exceptions::NumberFormatException::NumberFormatException (const std::exception * cause) throw () [inline]`

Constructor.

Parameters

***cause* Pointer** (p. 2343) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.498.1.6 `decaf::lang::exceptions::NumberFormatException::NumberFormatException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

References `decaf::lang::Exception::buildMessage()`, and `decaf::lang::Exception::setMark()`.

6.498.1.7 `virtual decaf::lang::exceptions::NumberFormatException::~~NumberFormatException () throw () [inline, virtual]`

6.498.2 Member Function Documentation

6.498.2.1 `virtual NumberFormatException* decaf::lang::exceptions::NumberFormatException::clone () const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

an new **Exception** (p. 1476) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1479).

References `NumberFormatException()`.

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/exceptions/NumberFormatException.h`

6.499 cms::ObjectMessage Class Reference

Place holder for interaction with JMS systems that support Java, the C++ client is not responsible for deserializing the contained Object.

```
#include <src/main/cms/ObjectMessage.h>
```

Inheritance diagram for cms::ObjectMessage:

Public Member Functions

- virtual `~ObjectMessage ()`

6.499.1 Detailed Description

Place holder for interaction with JMS systems that support Java, the C++ client is not responsible for deserializing the contained Object. serialized `ObjectMessage` (p. 2287)s.

Since

1.0

6.499.2 Constructor & Destructor Documentation

6.499.2.1 virtual cms::ObjectMessage::~ObjectMessage () [inline, virtual]

The documentation for this class was generated from the following file:

- `src/main/cms/ObjectMessage.h`

6.500 decaf::security_provider::unix::openssl::OpenSSLX500Principal Class Reference

The `OpenSSLX500Principal` (p. 2287) wraps around an OpenSSL `X509_NAME` structure.

```
#include <src/main/decaf/security_provider/unix/openssl/OpenSSLX500Principal.h>
```

Public Member Functions

- `OpenSSLX500Principal (X509_NAME *name)`
Constructor.
- virtual `~OpenSSLX500Principal ()`
Destructor.
- virtual `X509_NAME * getX509Name ()`
Accessor to the underlying X509 name structure.

- virtual bool **equals** (const Principal &another) const
Compares two principals to see if they are the same.
- virtual std::string **getName** () const
Returns the distinguished name string using the RFC2253 formatting.
- virtual void **getEncoded** (std::vector< unsigned char > &output) const
Serializes the distinguished name to its ASN.1 DER encoded form.

Static Public Member Functions

- static void **getEncoded** (X509_NAME *name, std::vector< unsigned char > &output)
Serializes the given distinguished name to its ASN.1 DER encoded form.
- static std::string **toString** (X509_NAME *name) const
Converts the given name to a string using the RFC2253 formatting.

6.500.1 Detailed Description

The `OpenSSLX500Principal` (p. 2287) wraps around an OpenSSL `X509_NAME` structure. It does not, however, control the lifetime of the structure.

6.500.2 Constructor & Destructor Documentation

6.500.2.1 `decaf::security_provider::unix::openssl::OpenSSLX500Principal::OpenSSLX500Principal (X509_NAME * name)`

Constructor.

Saves the internal X509 name and caches the string representation of the name.

Parameters

name The underlying X509 name structure.

6.500.2.2 `virtual decaf::security_provider::unix::openssl::OpenSSLX500Principal::~~OpenSSLX500Principal () [inline, virtual]`

Destructor.

Does nothing.

6.500.3 Member Function Documentation

6.500.3.1 `virtual bool decaf::security_-
provider::unix::openssl::OpenSSLX500Principal::equals (
const Principal & another) const [virtual]`

Compares two principals to see if they are the same.

Parameters

another A principal to be tested for equality to this one.

Returns

true if the given principal is equivalent to this one.

6.500.3.2 `virtual void decaf::security_-
provider::unix::openssl::OpenSSLX500Principal::getEncoded (
std::vector< unsigned char > & output) const [inline, virtual]`

Serializes the distinguished name to its ASN.1 DER encoded form.

Parameters

output Receives the distinguished name in ASN.1 DER encoded form.

6.500.3.3 `static void decaf::security_-
provider::unix::openssl::OpenSSLX500Principal::getEncoded (
X509_NAME * name, std::vector< unsigned char > & output)
[static]`

Serializes the given distinguished name to its ASN.1 DER encoded form.

Parameters

name the X509 name structure to be encoded.

output Receives the distinguished name in ASN.1 DER encoded form.

6.500.3.4 `virtual std::string decaf::security_-
provider::unix::openssl::OpenSSLX500Principal::getName
() const [inline, virtual]`

Returns the distinguished name string using the RFC2253 formatting.

Returns

the RFC2253 formatted distinguished name string.

References toString().

6.500.3.5 `virtual X509_NAME* decaf::security_provider::unix::openssl::OpenSSLX500Principal::getX509Name ()`
 [inline, virtual]

Accessor to the underlying X509 name structure.

6.500.3.6 `static std::string decaf::security_provider::unix::openssl::OpenSSLX500Principal::toString (`
 `X509_NAME * name) const [static]`

Converts the given name to a string using the RFC2253 formatting.

Parameters

name the X509 name structure to be formatted.

Returns

the RFC2253 formatted name string.

Referenced by getName().

The documentation for this class was generated from the following file:

- src/main/decaf/security_provider/unix/openssl/OpenSSLX500Principal.h

6.501 decaf::security_provider::unix::openssl::OpenSSLX509Certificate Class Reference

```
#include <src/main/decaf/security_provider/unix/openssl/OpenSSLX509Certificate.h>
```

Inheritance diagram for decaf::security_provider::unix::openssl::OpenSSLX509Certificate:

Public Member Functions

- virtual `~OpenSSLX509Certificate ()`
- virtual bool **equals** (const Certificate &cert) const =0
Compares the encoded form of the two certificates.
- virtual void **getEncoded** (std::vector< unsigned char > &output) const =0 throw (CertificateEncodingException)
Provides the encoded form of this certificate.
- virtual std::string **getType** () const =0
Returns the type of this certificate.
- virtual PublicKey * **getPublicKey** ()=0
Gets the public key of this certificate.

- virtual const PublicKey * **getPublicKey** () const =0
Gets the public key of this certificate.
- virtual void **verify** (const PublicKey &publicKey) const =0 throw (NoSuchAlgorithmException, InvalidKeyException, NoSuchProviderException, SignatureException, CertificateException)
Verifies that this certificate was signed with the private key that corresponds to the specified public key.
- virtual void **verify** (const PublicKey &publicKey, const std::string &sigProvider) const =0 throw (NoSuchAlgorithmException, InvalidKeyException, NoSuchProviderException, SignatureException, CertificateException)
Verifies that this certificate was signed with the private key that corresponds to the specified public key.
- virtual std::string **toString** () const =0
Returns a string representation of this certificate.
- virtual void **checkValidity** () const =0 throw (CertificateExpiredException, CertificateNotYetValidException)
- virtual void **checkValidity** (const decaf::util::Date &date) const =0 throw (CertificateExpiredException, CertificateNotYetValidException)
- virtual int **getBasicConstraints** () const =0
- virtual void **getIssuerUniqueID** (std::vector< bool > &output) const =0
- virtual const X500Principal * **getIssuerX500Principal** () const =0
- virtual void **getKeyUsage** (std::vector< unsigned char > &output) const =0
- virtual Date **getNotAfter** () const =0
- virtual Date **getNotBefore** () const =0
- virtual std::string **getSigAlgName** () const =0
- virtual std::string **getSigAlgOID** () const =0
- virtual void **getSigAlgParams** (std::vector< unsigned char > &output) const =0
- virtual void **getSignature** (std::vector< unsigned char > &output) const =0
- virtual void **getSubjectUniqueID** (std::vector< bool > &output) const =0
- virtual const X500Principal * **getSubjectX500Principal** () const =0
- virtual void **getTBSCertificate** (std::vector< unsigned char > &output) const =0 throw (CertificateEncodingException)
- virtual int **getVersion** () const =0

6.501.1 Constructor & Destructor Documentation

- 6.501.1.1 virtual decaf::security_-
provider::unix::openssl::OpenSSLX509Certificate::~~OpenSSLX509Certificate
() [virtual]

6.501.2 Member Function Documentation

- 6.501.2.1 virtual void decaf::security_-
provider::unix::openssl::OpenSSLX509Certificate::checkValidity
() const throw (CertificateExpiredException,
CertificateNotYetValidException) [pure virtual]

Implements decaf::security::cert::X509Certificate (p. 3190).

6.501.2.2 virtual void decaf::security_provider::unix::openssl::OpenSSLX509Certificate::checkValidity (const decaf::util::Date & *date*) const throw (CertificateExpiredException, CertificateNotYetValidException) [pure virtual]

Implements decaf::security::cert::X509Certificate (p. 3190).

6.501.2.3 virtual bool decaf::security_provider::unix::openssl::OpenSSLX509Certificate::equals (const Certificate & *cert*) const [pure virtual]

Compares the encoded form of the two certificates.

Parameters

cert The certificate to be tested for equality with this certificate.

Returns

true if the given certificate is equal to this certificate.

6.501.2.4 virtual int decaf::security_provider::unix::openssl::OpenSSLX509Certificate::getBasicConstraints () const [pure virtual]

Implements decaf::security::cert::X509Certificate (p. 3190).

6.501.2.5 virtual void decaf::security_provider::unix::openssl::OpenSSLX509Certificate::getEncoded (std::vector< unsigned char > & *output*) const throw (CertificateEncodingException) [pure virtual]

Provides the encoded form of this certificate.

Parameters

output Receives the encoded form of this certificate.

Exceptions

CertificateEncodingException if an encoding error occurs

Implements decaf::security::cert::Certificate (p. 902).

6.501.2.6 virtual void decaf::security_provider::unix::openssl::OpenSSLX509Certificate::getIssuerUniqueID (std::vector< bool > & *output*) const [pure virtual]

Implements decaf::security::cert::X509Certificate (p. 3190).

6.501.2.7 `virtual const X500Principal* decaf::security_-
provider::unix::openssl::OpenSSLX509Certificate::getIssuerX500Principal
() const [pure virtual]`

Implements `decaf::security::cert::X509Certificate` (p. 3191).

6.501.2.8 `virtual void decaf::security_-
provider::unix::openssl::OpenSSLX509Certificate::getKeyUsage (
std::vector< unsigned char > & output) const [pure virtual]`

Implements `decaf::security::cert::X509Certificate` (p. 3191).

6.501.2.9 `virtual Date decaf::security_-
provider::unix::openssl::OpenSSLX509Certificate::getNotAfter ()
const [pure virtual]`

Implements `decaf::security::cert::X509Certificate` (p. 3191).

6.501.2.10 `virtual Date decaf::security_-
provider::unix::openssl::OpenSSLX509Certificate::getNotBefore ()
const [pure virtual]`

Implements `decaf::security::cert::X509Certificate` (p. 3191).

6.501.2.11 `virtual PublicKey* decaf::security_-
provider::unix::openssl::OpenSSLX509Certificate::getPublicKey ()
[pure virtual]`

Gets the public key of this certificate.

Returns

the public key

Implements `decaf::security::cert::Certificate` (p. 902).

6.501.2.12 `virtual const PublicKey* decaf::security_-
provider::unix::openssl::OpenSSLX509Certificate::getPublicKey ()
const [pure virtual]`

Gets the public key of this certificate.

Returns

the public key

Implements `decaf::security::cert::Certificate` (p. 903).

6.501.2.13 virtual std::string decaf::security_provider::unix::openssl::OpenSSLX509Certificate::getSigAlgName ()
const [pure virtual]

Implements decaf::security::cert::X509Certificate (p. 3191).

6.501.2.14 virtual std::string decaf::security_provider::unix::openssl::OpenSSLX509Certificate::getSigAlgOID ()
const [pure virtual]

Implements decaf::security::cert::X509Certificate (p. 3191).

6.501.2.15 virtual void decaf::security_provider::unix::openssl::OpenSSLX509Certificate::getSigAlgParams (std::vector< unsigned char > & *output*) const [pure virtual]

Implements decaf::security::cert::X509Certificate (p. 3191).

6.501.2.16 virtual void decaf::security_provider::unix::openssl::OpenSSLX509Certificate::getSignature (std::vector< unsigned char > & *output*) const [pure virtual]

Implements decaf::security::cert::X509Certificate (p. 3191).

6.501.2.17 virtual void decaf::security_provider::unix::openssl::OpenSSLX509Certificate::getSubjectUniqueID (std::vector< bool > & *output*) const [pure virtual]

Implements decaf::security::cert::X509Certificate (p. 3192).

6.501.2.18 virtual const X500Principal* decaf::security_provider::unix::openssl::OpenSSLX509Certificate::getSubjectX500Principal () const [pure virtual]

Implements decaf::security::cert::X509Certificate (p. 3192).

6.501.2.19 virtual void decaf::security_provider::unix::openssl::OpenSSLX509Certificate::getTBSCertificate (std::vector< unsigned char > & *output*) const throw (CertificateEncodingException) [pure virtual]

Implements decaf::security::cert::X509Certificate (p. 3192).

6.501.2.20 virtual std::string decaf::security_provider::unix::openssl::OpenSSLX509Certificate::getType () const
[pure virtual]

Returns the type of this certificate.

Returns

the type of this certificate

Implements **decaf::security::cert::Certificate** (p. 903).

6.501.2.21 `virtual int decaf::security_
provider::unix::openssl::OpenSSLX509Certificate::getVersion () const
[pure virtual]`

Implements **decaf::security::cert::X509Certificate** (p. 3192).

6.501.2.22 `virtual std::string decaf::security_
provider::unix::openssl::OpenSSLX509Certificate::toString () const
[pure virtual]`

Returns a string representation of this certificate.

Returns

a string representation of this certificate

Implements **decaf::security::cert::Certificate** (p. 903).

6.501.2.23 `virtual void decaf::security_
provider::unix::openssl::OpenSSLX509Certificate::verify (
const PublicKey & publicKey) const throw (NoSuchAlgorith-
mException, InvalidKeyException, NoSuchProviderException,
SignatureException, CertificateException) [pure virtual]`

Verifies that this certificate was signed with the private key that corresponds to the specified public key.

Parameters

publicKey The public key used to carry out the validation.

Exceptions

NoSuchAlgorithmException - on unsupported signature algorithms.

InvalidKeyException - on incorrect key.

NoSuchProviderException - if there's no default provider.

SignatureException - on signature errors.

CertificateException - on encoding errors.

```

6.501.2.24 virtual void decaf::security_ -
              provider::unix::openssl::OpenSSLX509Certificate::verify (
                const PublicKey & publicKey, const std::string & sigProvider )
                const throw ( NoSuchAlgorithmException, InvalidKeyException,
                              NoSuchProviderException, SignatureException, CertificateException)
                [pure virtual]

```

Verifies that this certificate was signed with the private key that corresponds to the specified public key.

Uses the verification engine of the specified provider.

Parameters

publicKey The public key used to carry out the validation.
sigProvider The name of the signature provider

Exceptions

NoSuchAlgorithmException - on unsupported signature algorithms.
InvalidKeyException - on incorrect key.
NoSuchProviderException - if there's no default provider.
SignatureException - on signature errors.
CertificateException - on encoding errors.

The documentation for this class was generated from the following file:

- src/main/decaf/security_provider/unix/openssl/OpenSSLX509Certificate.h

6.502 activemq::wireformat::openwire::OpenWireFormat Class Reference

```
#include <src/main/activemq/wireformat/openwire/OpenWireFormat.h>
```

Inheritance diagram for activemq::wireformat::openwire::OpenWireFormat:

Public Member Functions

- **OpenWireFormat** (const decaf::util::Properties &properties)
Constructs a new OpenWireFormat (p. 2296) object.
- virtual ~OpenWireFormat ()
- virtual bool hasNegotiator () const
Returns true if this WireFormat (p. 3148) has a Negotiator that needs to wrap the Transport that uses it.
- virtual Pointer< transport::Transport > createNegotiator
(const Pointer< transport::Transport > &transport) throw (decaf::lang::exceptions::UnsupportedOperationException)

If the Transport Provides a Negotiator this method will create and return a news instance of the Negotiator.

- void **addMarshaller** (**marshal::DataStreamMarshaller** *marshaller)
Allows an external source to add marshalers to this object for types that may be marshaled or unmarshaled.
- virtual void **marshal** (const **Pointer**< **commands::Command** > &command, const **activemq::transport::Transport** *transport, **decaf::io::DataOutputStream** *out) throw (**decaf::io::IOException**)
Stream based marshaling of a Command, this method blocks until the entire Command has been written out to the output stream.
- virtual **Pointer**< **commands::Command** > **unmarshal** (const **activemq::transport::Transport** *transport, **decaf::io::DataInputStream** *in) throw (**decaf::io::IOException**)
Stream based un-marshaling, blocks on reads on the input stream until a complete command has been read and un-marshaled into the correct form.
- virtual int **tightMarshalNestedObject1** (**commands::DataStructure** *object, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Utility method for Tight Marshaling the given object to the boolean stream passed.
- void **tightMarshalNestedObject2** (**commands::DataStructure** *o, **decaf::io::DataOutputStream** *ds, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Utility method that will Tight marshal some internally nested object that implements the DataStructure interface.
- **commands::DataStructure** * **tightUnmarshalNestedObject** (**decaf::io::DataInputStream** *dis, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Utility method used to Unmarshal a Nested DataStructure type object from the given DataInputStream.
- **commands::DataStructure** * **looseUnmarshalNestedObject** (**decaf::io::DataInputStream** *dis) throw (**decaf::io::IOException**)
Utility method to unmarshal an DataStructure object from an DataInputStream using the Loose Unmarshaling format.
- void **looseMarshalNestedObject** (**commands::DataStructure** *o, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Utility method to loosely Marshal an object that is derived from the DataStrcutre interface.
- void **renegotiateWireFormat** (const **commands::WireFormatInfo** &info) throw (**decaf::lang::exceptions::IllegalStateException**)
Called to re-negotiate the settings for the WireFormatInfo, these determine how the client and broker communicate.
- virtual void **setPreferredWireFormatInfo** (const **Pointer**< **commands::WireFormatInfo** > &info) throw (**decaf::lang::exceptions::IllegalStateException**)

Configures this object using the provided WireFormatInfo object.

- virtual const **Pointer< commands::WireFormatInfo > & getPreferredWireFormatInfo** () const

Gets the Preferred WireFormatInfo object that this class holds.

- bool **isStackTraceEnabled** () const

Checks if the stackTraceEnabled flag is on.

- void **setStackTraceEnabled** (bool stackTraceEnabled)

Sets if the stackTraceEnabled flag is on.

- bool **isTcpNoDelayEnabled** () const

Checks if the tcpNoDelayEnabled flag is on.

- void **setTcpNoDelayEnabled** (bool tcpNoDelayEnabled)

Sets if the tcpNoDelayEnabled flag is on.

- int **getVersion** () const

Get the current Wireformat Version.

- void **setVersion** (int version) throw (decaf::lang::exceptions::IllegalArgumentException)

Set the current Wireformat Version.

- virtual bool **inReceive** () const

Is there a Message being unmarshaled?

- bool **isCacheEnabled** () const

Checks if the cacheEnabled flag is on.

- void **setCacheEnabled** (bool cacheEnabled)

Sets if the cacheEnabled flag is on.

- int **getCacheSize** () const

Returns the currently set Cache size.

- void **setCacheSize** (int value)

Sets the current Cache size.

- bool **isTightEncodingEnabled** () const

Checks if the tightEncodingEnabled flag is on.

- void **setTightEncodingEnabled** (bool tightEncodingEnabled)

Sets if the tightEncodingEnabled flag is on.

- bool **isSizePrefixDisabled** () const

Checks if the sizePrefixDisabled flag is on.

- void **setSizePrefixDisabled** (bool sizePrefixDisabled)

Sets if the sizePrefixDisabled flag is on.

- long long **getMaxInactivityDuration** () const
Gets the MaxInactivityDuration setting.
- void **setMaxInactivityDuration** (long long value)
Sets the MaxInactivityDuration setting.
- long long **getMaxInactivityDurationInitialDelay** () const
Gets the MaxInactivityDurationInitialDelay setting.
- void **setMaxInactivityDurationInitialDelay** (long long value)
Sets the MaxInactivityDurationInitialDelay setting.

Protected Member Functions

- **commands::DataStructure * doUnmarshal** (decaf::io::DataInputStream *dis)
throw (decaf::io::IOException)
Perform the actual unmarshal of data from the given DataInputStream return the unmarshalled DataStrucutre object once done, caller takes ownership of this object.
- void **destroyMarshallers** ()
Cleans up all registered Marshallers and empties the dataMarshallers vector.

Static Protected Attributes

- static const unsigned char **NULL_TYPE**
- static const int **DEFAULT_VERSION** = 1

6.502.1 Constructor & Destructor Documentation

6.502.1.1 activemq::wireformat::openwire::OpenWireFormat::OpenWireFormat (const decaf::util::Properties & *properties*)

Constructs a new **OpenWireFormat** (p. 2296) object.

Parameters

properties - can contain optional config params.

6.502.1.2 virtual
activemq::wireformat::openwire::OpenWireFormat::~~OpenWireFormat () [virtual]

6.502.2 Member Function Documentation

6.502.2.1 void activemq::wireformat::openwire::OpenWireFormat::addMarshaller (marshal::DataStreamMarshaller * *marshaller*)

Allows an external source to add marshalers to this object for types that may be marshaled or unmarshaled.

Parameters

marshaller - the Marshaller to add to the collection.

6.502.2.2 virtual Pointer<transport::Transport> activemq::wireformat::openwire::OpenWireFormat::createNegotiator (const Pointer< transport::Transport > & *transport*) throw (decaf::lang::exceptions::UnsupportedOperationException) [virtual]

If the Transport Provides a Negotiator this method will create and return a news instance of the Negotiator.

Returns

new instance of a **WireFormatNegotiator** (p. 3182).

Implements **activemq::wireformat::WireFormat** (p. 3149).

6.502.2.3 void activemq::wireformat::openwire::OpenWireFormat::destroyMarshallers () [protected]

Cleans up all registered Marshallers and empties the dataMarshallers vector.

This should be called before a reconfiguration of the version marshallers, or on destruction of this object

6.502.2.4 commands::DataStructure* activemq::wireformat::openwire::OpenWireFormat::doUnmarshal (decaf::io::DataInputStream * *dis*) throw (decaf::io::IOException) [protected]

Perform the actual unmarshal of data from the given DataInputStream return the unmarshalled DataStrucutre object once done, caller takes ownership of this object.

This method can return null if the type of the object to unmarshal is NULL, empty data.

Parameters

dis - DataInputStream to read from

Returns

new DataStructure* that the caller owns

Exceptions

IOException if an error occurs during the unmarshal

6.502.2.5 `int activemq::wireformat::openwire::OpenWireFormat::getCacheSize ()
const [inline]`

Returns the currently set Cache size.

Returns

the current value of the broker's cache size.

6.502.2.6 `long long ac-
tivemq::wireformat::openwire::OpenWireFormat::getMaxInactivityDuration
() const [inline]`

Gets the MaxInactivityDuration setting.

Returns

maximum inactivity duration value in milliseconds.

6.502.2.7 `long long ac-
tivemq::wireformat::openwire::OpenWireFormat::getMaxInactivityDurationInitialDelay
() const [inline]`

Gets the MaxInactivityDurationInitialDelay setting.

Returns

maximum inactivity duration initial delay value in milliseconds.

6.502.2.8 `virtual const Pointer<commands::WireFormatInfo>& ac-
tivemq::wireformat::openwire::OpenWireFormat::getPreferredWireFormatInfo
() const [inline, virtual]`

Gets the Preferred WireFormatInfo object that this class holds.

Returns

pointer to a preferred WireFormatInfo object

6.502.2.9 `int activemq::wireformat::openwire::OpenWireFormat::getVersion () const [inline, virtual]`

Get the current Wireformat Version.

Returns

int that identifies the version

Implements **activemq::wireformat::WireFormat** (p. 3149).

6.502.2.10 `virtual bool activemq::wireformat::openwire::OpenWireFormat::hasNegotiator () const [inline, virtual]`

Returns true if this **WireFormat** (p. 3148) has a Negotiator that needs to wrap the Transport that uses it.

Returns

true if the **WireFormat** (p. 3148) provides a Negotiator.

Implements **activemq::wireformat::WireFormat** (p. 3149).

6.502.2.11 `virtual bool activemq::wireformat::openwire::OpenWireFormat::inReceive () const [inline, virtual]`

Is there a Message being unmarshaled?

Returns

true while in the doUnmarshal method.

Implements **activemq::wireformat::WireFormat** (p. 3150).

6.502.2.12 `bool activemq::wireformat::openwire::OpenWireFormat::isCacheEnabled () const [inline]`

Checks if the cacheEnabled flag is on.

Returns

true if the flag is on.

6.502.2.13 `bool activemq::wireformat::openwire::OpenWireFormat::isSizePrefixDisabled () const [inline]`

Checks if the sizePrefixDisabled flag is on.

Returns

true if the flag is on.

6.502.2.14 `bool activemq::wireformat::openwire::OpenWireFormat::isStackTraceEnabled () const [inline]`

Checks if the `isStackTraceEnabled` flag is on.

Returns

true if the flag is on.

6.502.2.15 `bool activemq::wireformat::openwire::OpenWireFormat::isTcpNoDelayEnabled () const [inline]`

Checks if the `isTcpNoDelayEnabled` flag is on.

Returns

true if the flag is on.

6.502.2.16 `bool activemq::wireformat::openwire::OpenWireFormat::isTightEncodingEnabled () const [inline]`

Checks if the `isTightEncodingEnabled` flag is on.

Returns

true if the flag is on.

6.502.2.17 `void activemq::wireformat::openwire::OpenWireFormat::looseMarshalNestedObject (commands::DataStructure * o, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)`

Utility method to loosely Marshal an object that is derived from the `DataStructure` interface.

The marshaled data is written to the passed in `DataOutputStream`.

Parameters

o - `DataStructure` derived Object to Marshal

dataOut - `DataOutputStream` to write the data to

Exceptions

IOException if an error occurs.

6.502.2.18 `commands::DataStructure* activemq::wireformat::openwire::OpenWireFormat::looseUnmarshalNestedObject (decaf::io::DataInputStream * dis) throw (decaf::io::IOException)`

Utility method to unmarshal an DataStructure object from an DataInputStream using the Loose Unmarshaling format.

Will read the Data and construct a new DataStructure based Object, the pointer to the Object returned is now owned by the caller.

Parameters

dis - the DataInputStream to read the data from

Returns

a new DataStructure derived Object pointer

Exceptions

IOException if an error occurs.

6.502.2.19 `virtual void activemq::wireformat::openwire::OpenWireFormat::marshal (const Pointer< commands::Command > & command, const activemq::transport::Transport * transport, decaf::io::DataOutputStream * out) throw (decaf::io::IOException)`
[virtual]

Stream based marshaling of a Command, this method blocks until the entire Command has been written out to the output stream.

Parameters

command The Command to Marshal.

transport The Transport instance that called this method.

out The output stream to write the command to.

Exceptions

IOException

Implements `activemq::wireformat::WireFormat` (p. 3150).

6.502.2.20 `void activemq::wireformat::openwire::OpenWireFormat::renegotiateWireFormat (const commands::WireFormatInfo & info) throw (decaf::lang::exceptions::IllegalStateException)`

Called to re-negotiate the settings for the WireFormatInfo, these determine how the client and broker communicate.

Parameters

info - The new Wireformat Info settings

Exceptions

IllegalStateException is wire format can't be negotiated.

6.502.2.21 `void activemq::wireformat::openwire::OpenWireFormat::setCacheEnabled (bool cacheEnabled) [inline]`

Sets if the cacheEnabled flag is on.

Parameters

cacheEnabled - true to turn flag is on

6.502.2.22 `void activemq::wireformat::openwire::OpenWireFormat::setCacheSize (int value) [inline]`

Sets the current Cache size.

Parameters

value - the value to send as the broker's cache size.

6.502.2.23 `void activemq::wireformat::openwire::OpenWireFormat::setMaxInactivityDuration (long long value) [inline]`

Sets the MaxInactivityDuration setting.

Parameters

value - the Max inactivity duration value in milliseconds.

6.502.2.24 `void activemq::wireformat::openwire::OpenWireFormat::setMaxInactivityDurationInitialDelay (long long value) [inline]`

Sets the MaxInactivityDurationInitialDelay setting.

Parameters

value - the Max inactivity Initial Delay duration value in milliseconds.

6.502.2.25 `virtual void activemq::wireformat::openwire::OpenWireFormat::setPreferredWireFormatInfo (const Pointer< commands::WireFormatInfo > & info) throw (decaf::lang::exceptions::IllegalStateException) [virtual]`

Configures this object using the provided WireformatInfo object.

Parameters

info - a WireFormatInfo object, takes ownership.

6.502.2.26 void activemq::wireformat::openwire::OpenWireFormat::setSizePrefixDisabled (bool *sizePrefixDisabled*) [inline]

Sets if the sizePrefixDisabled flag is on.

Parameters

sizePrefixDisabled - true to turn flag is on

6.502.2.27 void activemq::wireformat::openwire::OpenWireFormat::setStackTraceEnabled (bool *stackTraceEnabled*) [inline]

Sets if the stackTraceEnabled flag is on.

Parameters

stackTraceEnabled - true to turn flag is on

6.502.2.28 void activemq::wireformat::openwire::OpenWireFormat::setTcpNoDelayEnabled (bool *tcpNoDelayEnabled*) [inline]

Sets if the tcpNoDelayEnabled flag is on.

Parameters

tcpNoDelayEnabled - true to turn flag is on

6.502.2.29 void activemq::wireformat::openwire::OpenWireFormat::setTightEncodingEnabled (bool *tightEncodingEnabled*) [inline]

Sets if the tightEncodingEnabled flag is on.

Parameters

tightEncodingEnabled - true to turn flag is on

6.502.2.30 void activemq::wireformat::openwire::OpenWireFormat::setVersion (int *version*) throw (decaf::lang::exceptions::IllegalArgumentException) [virtual]

Set the current Wireformat Version.

Parameters

version - int that identifies the version

Implements **activemq::wireformat::WireFormat** (p. 3150).

6.502.2.31 `virtual int activemq::wireformat::openwire::OpenWireFormat::tightMarshalNestedObject1 (commands::DataStructure * object, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Utility method for Tight Marshaling the given object to the boolean stream passed.

Parameters

object - The DataStructure to marshal
bs - the BooleanStream to write to

Returns

size of the data returned.

6.502.2.32 `void activemq::wireformat::openwire::OpenWireFormat::tightMarshalNestedObject2 (commands::DataStructure * o, decaf::io::DataOutputStream * ds, utils::BooleanStream * bs) throw (decaf::io::IOException)`

Utility method that will Tight marshal some internally nested object that implements the DataStructure interface.

Writes the data to the Data Output Stream provided.

Parameters

o - DataStructure object
ds - DataOutputStream for writing
bs - BooleanStream

Exceptions

IOException if an error occurs.

6.502.2.33 `commands::DataStructure* activemq::wireformat::openwire::OpenWireFormat::tightUnmarshalNestedObject (decaf::io::DataInputStream * dis, utils::BooleanStream * bs) throw (decaf::io::IOException)`

Utility method used to Unmarshal a Nested DataStructure type object from the given DataInputStream.

The DataStructure instance that is returned is now the property of the caller.

Parameters

dis - DataInputStream to read from
bs - BooleanStream to read from

Returns

Newly allocated DataStructure Object

Exceptions

IOException if an error occurs.

6.502.2.34 virtual Pointer<commands::Command> activemq::wireformat::openwire::OpenWireFormat::unmarshal (const activemq::transport::Transport * *transport*, decaf::io::DataInputStream * *in*) throw (decaf::io::IOException) [virtual]

Stream based un-marshaling, blocks on reads on the input stream until a complete command has been read and un-marshaled into the correct form.

Returns a Pointer to the newly un-marshaled Command.

Parameters

transport - Pointer to the transport that is making this request.

in - the input stream to read the command from.

Returns

the newly marshaled Command, caller owns the pointer

Exceptions

IOException

Implements **activemq::wireformat::WireFormat** (p. 3151).

6.502.3 Field Documentation

6.502.3.1 const int
activemq::wireformat::openwire::OpenWireFormat::DEFAULT_VERSION = 1 [static, protected]

6.502.3.2 const unsigned char
activemq::wireformat::openwire::OpenWireFormat::NULL_TYPE [static, protected]

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/OpenWireFormat.h

6.503 activemq::wireformat::openwire::OpenWireFormatFactory Class Reference

```
#include <src/main/activemq/wireformat/openwire/OpenWireFormatFactory.h>
```

Inheritance diagram for activemq::wireformat::openwire::OpenWireFormatFactory:

Public Member Functions

- **OpenWireFormatFactory** ()

Constructor - Sets Defaults for all properties, these are all subject to change once the `createWireFormat` method is called.

- virtual **~OpenWireFormatFactory** ()

- virtual **Pointer< wireformat::WireFormat > createWireFormat** (const **decaf::util::Properties** &properties) throw (**decaf::lang::exceptions::IllegalStateException**)

*Creates a new **WireFormat** (p. 3148) Object passing it a set of properties from which it can obtain any optional settings.*

6.503.1 Constructor & Destructor Documentation

6.503.1.1 **activemq::wireformat::openwire::OpenWireFormatFactory::OpenWireFormatFactory** () [inline]

Constructor - Sets Defaults for all properties, these are all subject to change once the `createWireFormat` method is called.

URL options ----- `wireFormat.stackTraceEnabled` `wireFormat.cacheEnabled` `wireFormat.tcpNoDelayEnabled` `wireFormat.tightEncodingEnabled` `wireFormat.sizePrefixDisabled` `wireFormat.maxInactivityDuration` `wireFormat.maxInactivityDurationInitialDelay`

6.503.1.2 **virtual** **activemq::wireformat::openwire::OpenWireFormatFactory::~~OpenWireFormatFactory** () [inline, virtual]

6.503.2 Member Function Documentation

6.503.2.1 **virtual Pointer<wireformat::WireFormat> activemq::wireformat::openwire::OpenWireFormatFactory::createWireFormat** (const **decaf::util::Properties** & *properties*) throw (**decaf::lang::exceptions::IllegalStateException**) [virtual]

Creates a new **WireFormat** (p. 3148) Object passing it a set of properties from which it can obtain any optional settings.

Parameters

properties - the Properties for this **WireFormat** (p. 3148)

Implements **activemq::wireformat::WireFormatFactory** (p. 3152).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/OpenWireFormatFactory.h`

6.504 activemq::wireformat::openwire::OpenWireFormatNegotiator Class Reference

```
#include <src/main/activemq/wireformat/openwire/OpenWireFormatNegotiator.h>
```

Inheritance diagram for activemq::wireformat::openwire::OpenWireFormatNegotiator:

Public Member Functions

- **OpenWireFormatNegotiator** (**OpenWireFormat** *wireFormat, const **Pointer**< **transport::Transport** > &next)
Constructor - Initializes this object around another Transport.
- virtual **~OpenWireFormatNegotiator** ()
- virtual void **oneway** (const **Pointer**< **commands::Command** > &command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)
Sends a one-way command.
- virtual **Pointer**< **commands::Response** > **request** (const **Pointer**< **commands::Command** > &command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)
Sends the given request to the server and waits for the response.
- virtual **Pointer**< **commands::Response** > **request** (const **Pointer**< **commands::Command** > &command, unsigned int timeout) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)
Sends the given request to the server and waits for the response.
- virtual void **onCommand** (const **Pointer**< **commands::Command** > &command)
This is called in the context of the nested transport's reading thread.
- virtual void **onTransportException** (**transport::Transport** *source, const **decaf::lang::Exception** &ex)
Event handler for an exception from a command transport.
- virtual void **start** () throw (decaf::io::IOException)
Starts this transport object and creates the thread for polling on the input stream for commands.
- virtual void **close** () throw (decaf::io::IOException)
Stops the polling thread and closes the streams.

6.504.1 Constructor & Destructor Documentation

- 6.504.1.1** **activemq::wireformat::openwire::OpenWireFormatNegotiator::OpenWireFormatNegotiator** (**OpenWireFormat** * wireFormat, const **Pointer**< **transport::Transport** > & next)

Constructor - Initializes this object around another Transport.

Parameters

wireFormat - The **WireFormat** (p.3148) object we use to negotiate

next - The next transport in the chain

6.504.1.2 virtual

activemq::wireformat::openwire::OpenWireFormatNegotiator::~~OpenWireFormatNegotiator () [virtual]

6.504.2 Member Function Documentation**6.504.2.1 virtual void ac-**

tivemq::wireformat::openwire::OpenWireFormatNegotiator::close ()
throw (decaf::io::IOException) [virtual]

Stops the polling thread and closes the streams.

This can be called explicitly, but is also called in the destructor. Once this object has been closed, it cannot be restarted.

Exceptions

IOException if errors occur.

Reimplemented from **activemq::transport::TransportFilter** (p.3075).

6.504.2.2 virtual void ac-

tivemq::wireformat::openwire::OpenWireFormatNegotiator::onCommand
(const Pointer< commands::Command > & command) [virtual]

This is called in the context of the nested transport's reading thread.

In the case of a response object, updates the request map and notifies those waiting on the response. Non-response messages are just delegated to the command listener.

Parameters

command the received from the nested transport.

Reimplemented from **activemq::transport::TransportFilter** (p.3077).

6.504.2.3 virtual void ac-

tivemq::wireformat::openwire::OpenWireFormatNegotiator::oneway
(const Pointer< commands::Command > &
command) throw (decaf::io::IOException, de-
caf::lang::exceptions::UnsupportedOperationException)
 [virtual]

Sends a one-way command.

Does not wait for any response from the broker. First waits for the WireFormatInfo exchange to happen so that we know how to encode out-bound data.

Parameters

command the command to be sent.

Exceptions

IOException if an exception occurs during writing of the command.

UnsupportedOperationException if this method is not implemented by this transport.

Reimplemented from **activemq::transport::TransportFilter** (p. 3077).

6.504.2.4 `virtual void activemq::wireformat::openwire::OpenWireFormatNegotiator::onTransportException (transport::Transport * source, const decaf::lang::Exception & ex) [virtual]`

Event handler for an exception from a command transport.

Parameters

source The source of the exception

ex The exception.

6.504.2.5 `virtual Pointer<commands::Response> activemq::wireformat::openwire::OpenWireFormatNegotiator::request (const Pointer< commands::Command > & command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Sends the given request to the server and waits for the response.

First waits for the WireFormatInfo exchange to happen so that we know how to encode out-bound data.

Parameters

command The request to send.

Returns

the response from the server.

Exceptions

IOException if an error occurs with the request.

Reimplemented from **activemq::transport::TransportFilter** (p. 3079).

6.504.2.6 `virtual Pointer<commands::Response> activemq::wireformat::openwire::OpenWireFormatNegotiator::request (const Pointer< commands::Command > & command, unsigned int timeout) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Sends the given request to the server and waits for the response.

First waits for the WireFormatInfo exchange to happen so that we know how to encode out-bound data.

Parameters

- command* The request to send.
- timeout* The time to wait for the response.

Returns

the response from the server.

Exceptions

- IOException* if an error occurs with the request.

Reimplemented from **activemq::transport::TransportFilter** (p. 3078).

```
6.504.2.7 virtual void ac-
          tivemq::wireformat::openwire::OpenWireFormatNegotiator::start ( )
          throw ( decaf::io::IOException ) [virtual]
```

Starts this transport object and creates the thread for polling on the input stream for commands.

If this object has been closed, throws an exception. Before calling start, the caller must set the IO streams and the reader and writer objects.

Exceptions

- IOException* if an error occurs or if this transport has already been closed.

Reimplemented from **activemq::transport::TransportFilter** (p. 3079).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/OpenWireFormatNegotiator.h`

6.505 activemq::wireformat::openwire::OpenWireResponseBuilder Class Reference

```
#include <src/main/activemq/wireformat/openwire/OpenWireResponseBuilder.h>
```

Inheritance diagram for `activemq::wireformat::openwire::OpenWireResponseBuilder`:

Public Member Functions

- **OpenWireResponseBuilder** ()
- virtual **~OpenWireResponseBuilder** ()
- virtual **Pointer< commands::Response > buildResponse** (const **Pointer< commands::Command >** &command)

Given a Command, check if it requires a response and return the appropriate Response that the Broker would send for this Command.

- virtual void **buildIncomingCommands** (const Pointer< commands::Command > &command, decaf::util::StlQueue< Pointer< commands::Command > > &queue)

When called the ResponseBuilder must construct all the Responses or Asynchronous commands that would be sent to this client by the Broker upon receipt of the passed command.

6.505.1 Constructor & Destructor Documentation

6.505.1.1 `activemq::wireformat::openwire::OpenWireResponseBuilder::OpenWireResponseBuilder () [inline]`

6.505.1.2 `virtual
activemq::wireformat::openwire::OpenWireResponseBuilder::~~OpenWireResponseBuilder () [inline, virtual]`

6.505.2 Member Function Documentation

6.505.2.1 `virtual void ac-
tivemq::wireformat::openwire::OpenWireResponseBuilder::buildIncomingCommands
(const Pointer< commands::Command > & command,
decaf::util::StlQueue< Pointer< commands::Command > > & queue)
[virtual]`

When called the ResponseBuilder must construct all the Responses or Asynchronous commands that would be sent to this client by the Broker upon receipt of the passed command.

Parameters

command - The Command being sent to the Broker.

queue - Queue of Command sent back from the broker.

Implements `activemq::transport::mock::ResponseBuilder` (p. 2615).

6.505.2.2 `virtual Pointer<commands::Response> ac-
tivemq::wireformat::openwire::OpenWireResponseBuilder::buildResponse
(const Pointer< commands::Command > & command) [virtual]`

Given a Command, check if it requires a response and return the appropriate Response that the Broker would send for this Command.

Parameters

command - The command to build a response for

Returns

A Response object pointer, or NULL if no response.

Implements `activemq::transport::mock::ResponseBuilder` (p. 2616).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/OpenWireResponseBuilder.h`

6.506 activemq::wireformat::openwire::utils::OpenwireStringSupport Class Reference

```
#include <src/main/activemq/wireformat/openwire/utils/OpenwireStringSupport.h>
```

Static Public Member Functions

- static std::string **readString** (decaf::io::DataInputStream &dataIn) throw (decaf::io::IOException)

Static method used for reading a string that uses the Openwire format from a DataInputStream, this can throw an IOException for the same reason as a DataInputStream.readUTF might, as well as if the string that is received doesn't conform to the supported character encoding.

- static void **writeString** (decaf::io::DataOutputStream &dataOut, const std::string *str) throw (decaf::io::IOException)

Static method used for writing a string that uses the Openwire format from a DataOutputStream, this can throw an IOException for the same reason as a DataOutputStream.writeUTF might.

Protected Member Functions

- OpenwireStringSupport ()
- virtual ~OpenwireStringSupport ()

6.506.1 Constructor & Destructor Documentation

6.506.1.1 activemq::wireformat::openwire::utils::OpenwireStringSupport::OpenwireStringSupport () [inline, protected]

6.506.1.2 virtual
activemq::wireformat::openwire::utils::OpenwireStringSupport::~~OpenwireStringSupport () [inline, protected, virtual]

6.506.2 Member Function Documentation

6.506.2.1 static std::string **activemq::wireformat::openwire::utils::OpenwireStringSupport::readString** (decaf::io::DataInputStream & *dataIn*) throw (decaf::io::IOException) [static]

Static method used for reading a string that uses the Openwire format from a DataInputStream, this can throw an IOException for the same reason as a DataInputStream.readUTF might, as well as if the string that is received doesn't conform to the supported character encoding.

Parameters

dataIn - DataInputStream to read from

Returns

A string that has been read

Exceptions

IOException on Error.

```

6.506.2.2 static void activemq::wireformat::openwire::utils::OpenwireStringSupport::writeString
( decaf::io::DataOutputStream & dataOut, const std::string * str )
throw ( decaf::io::IOException ) [static]

```

Static method used for writing a string that uses the Openwire format from a DataOutputStream, this can throw an IOException for the same reason as a DataOutputStream.writeUTF might.

Parameters

dataOut - DataOutputStream to write to

str - A pointer to a string that should be written, NULL needs to be an option here as its written as -1.

Exceptions

IOException on Error.

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/utils/OpenwireStringSupport.h

6.507 decaf::io::OutputStream Class Reference

Base interface for an output stream.

```
#include <src/main/decaf/io/OutputStream.h>
```

Inheritance diagram for decaf::io::OutputStream:

Public Member Functions

- virtual **~OutputStream** ()
- virtual void **write** (unsigned char c)=0 throw (IOException)
Writes a single byte to the output stream.
- virtual void **write** (const std::vector< unsigned char > &buffer)=0 throw (IOException)
Writes an array of bytes to the output stream.
- virtual void **write** (const unsigned char *buffer, std::size_t offset, std::size_t len)=0 throw (IOException, lang::exceptions::NullPointerException)
Writes an array of bytes to the output stream.
- virtual void **flush** ()=0 throw (IOException)
Flushes any pending writes in this output stream.

6.507.1 Detailed Description

Base interface for an output stream.

6.507.2 Constructor & Destructor Documentation

6.507.2.1 `virtual decaf::io::OutputStream::~~OutputStream () [inline, virtual]`

6.507.3 Member Function Documentation

6.507.3.1 `virtual void decaf::io::OutputStream::flush () throw (IOException) [pure virtual]`

Flushes any pending writes in this output stream.

Exceptions

IOException (p. 1707)

Implemented in `decaf::internal::io::StandardErrorOutputStream` (p. 2814), `decaf::internal::io::StandardOutputStream` (p. 2827), `decaf::io::BufferedOutputStream` (p. 753), `decaf::io::ByteArrayOutputStream` (p. 838), `decaf::io::FilterOutputStream` (p. 1539), and `decaf::net::SocketOutputStream` (p. 2805).

6.507.3.2 `virtual void decaf::io::OutputStream::write (const std::vector< unsigned char > & buffer) throw (IOException) [pure virtual]`

Writes an array of bytes to the output stream.

Parameters

buffer The bytes to write.

Exceptions

IOException (p. 1707) thrown if an error occurs.

Implemented in `decaf::internal::io::StandardErrorOutputStream` (p. 2817), `decaf::internal::io::StandardOutputStream` (p. 2829), `decaf::io::BufferedOutputStream` (p. 754), `decaf::io::ByteArrayOutputStream` (p. 843), `decaf::io::DataOutputStream` (p. 1303), `decaf::io::FilterOutputStream` (p. 1542), and `decaf::net::SocketOutputStream` (p. 2808).

6.507.3.3 `virtual void decaf::io::OutputStream::write (const unsigned char * buffer, std::size_t offset, std::size_t len) throw (IOException, lang::exceptions::NullPointerException) [pure virtual]`

Writes an array of bytes to the output stream.

Parameters

buffer The array of bytes to write.

offset the position to start writing in buffer.

len The number of bytes from the buffer to be written.

Exceptions

IOException (p. 1707) thrown if an error occurs.

NullPointerException thrown if buffer is Null.

Implemented in `activemq::io::LoggingOutputStream` (p. 1920), `decaf::internal::io::StandardErrorOutputStream` (p. 2817), `decaf::internal::io::StandardOutputStream` (p. 2830), `decaf::io::BufferedOutputStream` (p. 754), `decaf::io::ByteArrayOutputStream` (p. 842), `decaf::io::DataOutputStream` (p. 1303), `decaf::io::FilterOutputStream` (p. 1543), and `decaf::net::SocketOutputStream` (p. 2809).

6.507.3.4 `virtual void decaf::io::OutputStream::write (unsigned char c) throw (IOException) [pure virtual]`

Writes a single byte to the output stream.

Parameters

c the byte.

Exceptions

IOException (p. 1707) thrown if an error occurs.

Implemented in `activemq::io::LoggingOutputStream` (p. 1919), `decaf::internal::io::StandardErrorOutputStream` (p. 2817), `decaf::internal::io::StandardOutputStream` (p. 2830), `decaf::io::BufferedOutputStream` (p. 754), `decaf::io::ByteArrayOutputStream` (p. 842), `decaf::io::DataOutputStream` (p. 1302), `decaf::io::FilterOutputStream` (p. 1542), and `decaf::net::SocketOutputStream` (p. 2808).

The documentation for this class was generated from the following file:

- `src/main/decaf/io/OutputStream.h`

6.508 activemq::commands::PartialCommand Class Reference

```
#include <src/main/activemq/commands/PartialCommand.h>
```

Inheritance diagram for `activemq::commands::PartialCommand`:

Public Member Functions

- `PartialCommand ()`

- virtual `~PartialCommand ()`
- virtual unsigned char `getDataStructureType ()` const

Get the unique identifier that this object and its own Marshaler share.

- virtual `PartialCommand * cloneDataStructure ()` const

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void `copyDataStructure (const DataStructure *src)`

Copy the contents of the passed object into this object's members, overwriting any existing data.

- virtual std::string `toString ()` const

*Returns a string containing the information for this **DataStructure** (p. 1372) such as its type and value of its elements.*

- virtual bool `equals (const DataStructure *value)` const

*Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent.*

- virtual int `getCommandId ()` const
- virtual void `setCommandId (int commandId)`
- virtual const std::vector< unsigned char > & `getData ()` const
- virtual std::vector< unsigned char > & `getData ()`
- virtual void `setData (const std::vector< unsigned char > &data)`

Static Public Attributes

- static const unsigned char `ID_PARTIALCOMMAND = 60`

Protected Member Functions

- `PartialCommand (const PartialCommand &)`
- `PartialCommand & operator= (const PartialCommand &)`

Protected Attributes

- int `commandId`
- std::vector< unsigned char > `data`

6.508.1 Constructor & Destructor Documentation

6.508.1.1 `activemq::commands::PartialCommand::PartialCommand (const PartialCommand &)` [inline, protected]

6.508.1.2 `activemq::commands::PartialCommand::PartialCommand ()`

6.508.1.3 `virtual activemq::commands::PartialCommand::~~PartialCommand ()` [virtual]

6.508.2 Member Function Documentation

6.508.2.1 `virtual PartialCommand* activemq::commands::PartialCommand::cloneDataStructure () const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1373).

Reimplemented in `activemq::commands::LastPartialCommand` (p. 1841).

6.508.2.2 `virtual void activemq::commands::PartialCommand::copyDataStructure (const DataStructure * src)` [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

Implements `activemq::commands::DataStructure` (p. 1374).

Reimplemented in `activemq::commands::LastPartialCommand` (p. 1841).

6.508.2.3 `virtual bool activemq::commands::PartialCommand::equals (const DataStructure * value) const` [virtual]

Compares the `DataStructure` (p. 1372) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Implements `activemq::commands::DataStructure` (p. 1374).

Reimplemented in `activemq::commands::LastPartialCommand` (p. 1841).

6.508.2.4 `virtual int activemq::commands::PartialCommand::getCommandId ()`
`const [virtual]`

6.508.2.5 `virtual const std::vector<unsigned char>& ac-`
`tivemq::commands::PartialCommand::getData () const`
`[virtual]`

6.508.2.6 `virtual std::vector<unsigned char>& ac-`
`tivemq::commands::PartialCommand::getData ()`
`[virtual]`

6.508.2.7 `virtual unsigned char ac-`
`tivemq::commands::PartialCommand::getDataStructureType () const`
`[virtual]`

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataStructure** (p. 1372) type copy.

Implements **activemq::commands::DataStructure** (p. 1375).

Reimplemented in **activemq::commands::LastPartialCommand** (p. 1842).

6.508.2.8 `PartialCommand& activemq::commands::PartialCommand::operator= (`
`const PartialCommand &) [inline, protected]`

6.508.2.9 `virtual void activemq::commands::PartialCommand::setCommandId (`
`int commandId) [virtual]`

6.508.2.10 `virtual void activemq::commands::PartialCommand::setData (const`
`std::vector< unsigned char > & data) [virtual]`

6.508.2.11 `virtual std::string activemq::commands::PartialCommand::toString ()`
`const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1372) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 662).

Reimplemented in **activemq::commands::LastPartialCommand** (p. 1842).

6.508.3 Field Documentation

- 6.508.3.1 `int activemq::commands::PartialCommand::commandId` [protected]
- 6.508.3.2 `std::vector<unsigned char> activemq::commands::PartialCommand::data`
[protected]
- 6.508.3.3 `const unsigned char activemq::commands::PartialCommand::ID_ - PARTIALCOMMAND = 60` [static]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/PartialCommand.h`

6.509 activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2322).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/PartialCommandMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller`:

Public Member Functions

- **PartialCommandMarshaller** ()
- virtual **~PartialCommandMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.

- virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.509.1 Detailed Description

Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2322). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.509.2 Constructor & Destructor Documentation

6.509.2.1 activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller::PartialCommandMarshaller () [inline]

6.509.2.2 virtual activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller::~~PartialCommandMarshaller () [inline, virtual]

6.509.3 Member Function Documentation

6.509.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1331).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller** (p. 1844).

6.509.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1336).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller** (p. 1844).

6.509.3.3 virtual void **activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller::looseMarshal** (**OpenWireFormat** * *wireFormat*, **commands::DataStructure** * *dataStructure*, **decaf::io::DataOutputStream** * *dataOut*) throw (**decaf::io::IOException**) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1342).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller** (p. 1844).

6.509.3.4 virtual void **activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller::looseUnmarshal** (**OpenWireFormat** * *wireFormat*, **commands::DataStructure** * *dataStructure*, **decaf::io::DataInputStream** * *dataIn*) throw (**decaf::io::IOException**) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1348).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller** (p. 1845).

```

6.509.3.5  virtual int ac-
            tivemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller::tightMarshal1
            ( OpenWireFormat * wireFormat, commands::DataStructure
            * dataStructure, utils::BooleanStream * bs ) throw (
            decaf::io::IOException ) [virtual]

```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1354).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller` (p. 1845).

```

6.509.3.6  virtual void ac-
            tivemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller::tightMarshal2
            ( OpenWireFormat * wireFormat, commands::DataStructure
            * dataStructure, decaf::io::DataOutputStream * dataOut,
            utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1360).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller` (p. 1846).

6.509.3.7 virtual void activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1366).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller** (p. 1846).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/PartialCommandMarshaller.h

6.510 activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2326).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/PartialCommandMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller:

Public Member Functions

- **PartialCommandMarshaller** ()
- virtual **~PartialCommandMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.510.1 Detailed Description

Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2326). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.510.2 Constructor & Destructor Documentation

6.510.2.1 activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller::PartialCommandMarshaller () [inline]

6.510.2.2 virtual
activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller::~~PartialCommandMarshaller () [inline, virtual]

6.510.3 Member Function Documentation

6.510.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1331).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller` (p. 1852).

6.510.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller::getDataStructureType() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller` (p. 1852).

6.510.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1342).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller` (p. 1852).

6.510.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1348).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller` (p. 1853).

```
6.510.3.5 virtual int ac-
tivemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller::tightMarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, utils::BooleanStream * bs ) throw (
decaf::io::IOException ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1354).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller` (p. 1853).

```
6.510.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1360).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller` (p. 1854).

6.510.3.7 virtual void `activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller::tightUnmarshal`
(`OpenWireFormat * wireFormat`, `commands::DataStructure`
* `dataStructure`, `decaf::io::DataInputStream * dataIn`,
`utils::BooleanStream * bs`) throw (`decaf::io::IOException`) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1366).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller` (p. 1854).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/PartialCommandMarshaller.h`

6.511 activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller Class Reference

Marshaling code for Open Wire Format for `PartialCommandMarshaller` (p. 2330).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/PartialCommandMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller`:

Public Member Functions

- `PartialCommandMarshaller` ()
- virtual `~PartialCommandMarshaller` ()

- virtual **commands::DataStructure * createObject ()** const

Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType ()** const

Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs)** throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs)** throw (decaf::io::IOException)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs)** throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn)** throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut)** throw (decaf::io::IOException)

Write a object instance to data output stream.

6.511.1 Detailed Description

Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2330). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.511.2 Constructor & Destructor Documentation

6.511.2.1 `activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller::PartialCommandMarshaller () [inline]`

6.511.2.2 `virtual activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller::~~PartialCommandMarshaller () [inline, virtual]`

6.511.3 Member Function Documentation

6.511.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller` (p. 1856).

6.511.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller` (p. 1856).

6.511.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1342).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller` (p. 1856).

6.511.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1348).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller` (p. 1857).

6.511.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1354).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller` (p. 1857).

6.511.3.6 virtual void activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1360).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller** (p. 1858).

6.511.3.7 virtual void activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1366).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller** (p. 1858).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/PartialCommandMarshaller.h

6.512 activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2335).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/PartialCommandMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller:

Public Member Functions

- **PartialCommandMarshaller** ()
- virtual **~PartialCommandMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.512.1 Detailed Description

Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2335). NOTE! This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.512.2 Constructor & Destructor Documentation

6.512.2.1 **activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller::PartialCommandMarshaller**
() [inline]

6.512.2.2 **virtual**
activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller::~~PartialCommandMarshaller
() [inline, virtual]

6.512.3 Member Function Documentation

6.512.3.1 **virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller::createObject**
() const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1331).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller** (p. 1848).

6.512.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller::getDataStructureType**
() const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1336).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller** (p. 1848).

6.512.3.3 **virtual void activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller::looseMarshal**
(**OpenWireFormat * wireFormat**, **commands::DataStructure * dataStructure**, **decaf::io::DataOutputStream * dataOut**) throw (**decaf::io::IOException**) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1342).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller` (p. 1848).

6.512.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1348).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller` (p. 1849).

6.512.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1354).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller** (p. 1849).

6.512.3.6 virtual void **activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller::tightMarshal2**
(**OpenWireFormat** * *wireFormat*, **commands::DataStructure**
* *dataStructure*, **decaf::io::DataOutputStream** * *dataOut*,
utils::BooleanStream * *bs*) throw (**decaf::io::IOException**) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1360).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller** (p. 1850).

6.512.3.7 virtual void **activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller::tightUnmarshal**
(**OpenWireFormat** * *wireFormat*, **commands::DataStructure**
* *dataStructure*, **decaf::io::DataInputStream** * *dataIn*,
utils::BooleanStream * *bs*) throw (**decaf::io::IOException**) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1366).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller** (p. 1850).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/PartialCommandMarshaller.h`

6.513 **activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller** Class Reference

Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2339).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/PartialCommandMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller**:

Public Member Functions

- **PartialCommandMarshaller** ()
- virtual **~PartialCommandMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.513.1 Detailed Description

Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2339). NOTE! This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.513.2 Constructor & Destructor Documentation

6.513.2.1 activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller::PartialCommandMarshaller () [inline]

6.513.2.2 virtual activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller::~~PartialCommandMarshaller () [inline, virtual]

6.513.3 Member Function Documentation

6.513.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1331).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller** (p. 1860).

6.513.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1336).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller** (p. 1860).

6.513.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1342).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller` (p. 1860).

6.513.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1348).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller` (p. 1861).

6.513.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1354).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller** (p. 1861).

```
6.513.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1360).

Reimplemented in **activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller** (p. 1862).

```
6.513.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1366).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller` (p. 1862).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/PartialCommandMarshaller.h`

6.514 decaf::lang::Pointer< T, REFCOUNTER > Class Template Reference

Decaf's implementation of a Smart **Pointer** (p. 2343) that is a template on a Type and is Thread Safe if the default Reference Counter is used.

```
#include <src/main/decaf/lang/Pointer.h>
```

Public Types

- `typedef T * PointerType`
- `typedef T & ReferenceType`
- `typedef REFCOUNTER CounterType`

Public Member Functions

- **Pointer** ()
Default Constructor.
- **Pointer** (const **PointerType** value)
*Explicit Constructor, creates a **Pointer** (p. 2343) that contains value with a single reference.*
- **Pointer** (const **Pointer** &value) throw ()
Copy constructor.
- `template<typename T1 , typename R1 >`
Pointer (const **Pointer**< T1, R1 > &value) throw ()
Copy constructor.
- `template<typename T1 , typename R1 >`
Pointer (const **Pointer**< T1, R1 > &value, const **STATIC_CAST_TOKEN** &) throw ()
Static Cast constructor.

- `template<typename T1 , typename R1 >`
`Pointer` (const `Pointer`< T1, R1 > &value, const `DYNAMIC_CAST_TOKEN` &)
throw (decaf::lang::exceptions::ClassCastException)
Dynamic Cast constructor.
- `virtual ~Pointer ()` throw ()
- `void reset (T *value)`
*Resets the **Pointer** (p. 2343) to hold the new value.*
- `T * release ()`
*Releases the **Pointer** (p. 2343) held and resets the internal pointer value to Null.*
- `PointerType get ()` const
*Gets the real pointer that is contained within this **Pointer** (p. 2343).*
- `void swap (Pointer &value)` throw ()
Exception (p. 1476) Safe Swap Function.
- `Pointer & operator= (const Pointer &right)` throw ()
*Assigns the value of right to this **Pointer** (p. 2343) and increments the reference Count.*
- `template<typename T1 , typename R1 >`
`Pointer & operator= (const Pointer< T1, R1 > &right)` throw ()
- `ReferenceType operator* ()`
Dereference Operator, returns a reference to the Contained value.
- `ReferenceType operator* ()` const
- `PointerType operator-> ()`
Indirection Operator, returns a pointer to the Contained value.
- `PointerType operator-> ()` const
- `bool operator! ()` const
- `template<typename T1 , typename R1 >`
`bool operator== (const Pointer< T1, R1 > &right)` const
- `template<typename T1 , typename R1 >`
`bool operator!= (const Pointer< T1, R1 > &right)` const
- `template<typename T1 >`
`Pointer< T1, CounterType > dynamicCast ()` const
- `template<typename T1 >`
`Pointer< T1, CounterType > staticCast ()` const

Friends

- `bool operator== (const Pointer &left, const T *right)`
- `bool operator== (const T *left, const Pointer &right)`
- `bool operator!= (const Pointer &left, const T *right)`
- `bool operator!= (const T *left, const Pointer &right)`

6.514.1 Detailed Description

```
template<typename T, typename REFCOUNTER = AtomicRefCount> class
decaf::lang::Pointer< T, REFCOUNTER >
```

Decaf's implementation of a Smart **Pointer** (p. 2343) that is a template on a Type and is Thread Safe if the default Reference Counter is used. This **Pointer** (p. 2343) type allows for the substitution of different Reference Counter implementations which provide a means of using invasive reference counting if desired using a custom implementation of **ReferenceCounter**.

The Decaf smart pointer provide comparison operators for comparing **Pointer** (p. 2343) instances in the same manner as normal pointer, except that it does not provide an overload of operators (<, <=, >, >=). To allow use of a **Pointer** (p. 2343) in a STL container that requires it, **Pointer** (p. 2343) provides an implementation of std::less.

Since

1.0

6.514.2 Member Typedef Documentation

6.514.2.1 `template<typename T, typename REFCOUNTER = AtomicRefCount> typedef REFCOUNTER decaf::lang::Pointer< T, REFCOUNTER >::CounterType`

6.514.2.2 `template<typename T, typename REFCOUNTER = AtomicRefCount> typedef T* decaf::lang::Pointer< T, REFCOUNTER >::PointerType`

6.514.2.3 `template<typename T, typename REFCOUNTER = AtomicRefCount> typedef T& decaf::lang::Pointer< T, REFCOUNTER >::ReferenceType`

6.514.3 Constructor & Destructor Documentation

6.514.3.1 `template<typename T, typename REFCOUNTER = AtomicRefCount> decaf::lang::Pointer< T, REFCOUNTER >::Pointer () [inline]`

Default Constructor.

Initialized the contained pointer to NULL, using the -> operator results in an exception unless reset to contain a real value.

Referenced by decaf::lang::Pointer< TransactionId >::reset().

6.514.3.2 `template<typename T, typename REFCOUNTER = AtomicRefCount> decaf::lang::Pointer< T, REFCOUNTER >::Pointer (const PointerType value) [inline, explicit]`

Explicit Constructor, creates a **Pointer** (p. 2343) that contains value with a single reference.

This object now has ownership until a call to release.

Parameters

value - instance of the type we are containing here.

6.514.3.3 `template<typename T, typename REFCOUNTER = AtomicRefCount> decaf::lang::Pointer< T, REFCOUNTER >::Pointer (const Pointer< T, REFCOUNTER > & value) throw () [inline]`

Copy constructor.

Copies the value contained in the pointer to the new instance and increments the reference counter.

6.514.3.4 `template<typename T, typename REFCOUNTER = AtomicRefCount> template<typename T1 , typename R1 > decaf::lang::Pointer< T, REFCOUNTER >::Pointer (const Pointer< T1, R1 > & value) throw () [inline]`

Copy constructor.

Copies the value contained in the pointer to the new instance and increments the reference counter.

6.514.3.5 `template<typename T, typename REFCOUNTER = AtomicRefCount> template<typename T1 , typename R1 > decaf::lang::Pointer< T, REFCOUNTER >::Pointer (const Pointer< T1, R1 > & value, const STATIC_CAST_TOKEN &) throw () [inline]`

Static Cast constructor.

Copies the value contained in the pointer to the new instance and increments the reference counter performing a static cast on the value contained in the source **Pointer** (p.2343) object.

Parameters

value - **Pointer** (p.2343) instance to cast to this type.

6.514.3.6 `template<typename T, typename REFCOUNTER = AtomicRefCount> template<typename T1 , typename R1 > decaf::lang::Pointer< T, REFCOUNTER >::Pointer (const Pointer< T1, R1 > & value, const DYNAMIC_CAST_TOKEN &) throw (decaf::lang::exceptions::ClassCastException) [inline]`

Dynamic Cast constructor.

Copies the value contained in the pointer to the new instance and increments the reference counter performing a dynamic cast on the value contained in the source **Pointer** (p.2343) object. If the cast fails and return NULL then this method throws a ClassCastException.

Parameters

value - **Pointer** (p.2343) instance to cast to this type.

Exceptions

ClassCastException if the dynamic cast returns NULL

```
6.514.3.7  template<typename T, typename REFCOUNTER =
           AtomicRefCount> virtual decaf::lang::Pointer< T, REFCOUNTER
           >::~~Pointer ( ) throw () [inline, virtual]
```

6.514.4 Member Function Documentation

```
6.514.4.1  template<typename T, typename REFCOUNTER =
           AtomicRefCount> template<typename T1 > Pointer<T1,
           CounterType> decaf::lang::Pointer< T, REFCOUNTER >::dynamicCast
           ( ) const [inline]
```

```
6.514.4.2  template<typename T, typename REFCOUNTER =
           AtomicRefCount> PointerType decaf::lang::Pointer< T,
           REFCOUNTER >::get ( ) const [inline]
```

Gets the real pointer that is contained within this **Pointer** (p. 2343).

This is not really safe since the caller could delete or alter the pointer but it mimics the STL `auto_ptr` and gives access in cases where the caller absolutely needs the real **Pointer** (p. 2343). Use at your own risk.

Returns

the contained pointer.

Referenced by `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::equals()`, `activemq::core::ActiveMQConnectionSupport::getProperties()`, `activemq::core::ActiveMQConnectionSupport::getTransport()`, `decaf::lang::operator!=()`, `decaf::lang::Pointer< TransactionId >::operator!=()`, `std::less< decaf::lang::Pointer< T > >::operator()()`, `decaf::lang::operator==()`, `decaf::lang::Pointer< TransactionId >::operator==()`, and `activemq::state::ConnectionState::removeTempDestination()`.

```
6.514.4.3  template<typename T, typename REFCOUNTER =
           AtomicRefCount> bool decaf::lang::Pointer< T, REFCOUNTER
           >::operator! ( ) const [inline]
```

```
6.514.4.4  template<typename T, typename REFCOUNTER =
           AtomicRefCount> template<typename T1 , typename R1 > bool
           decaf::lang::Pointer< T, REFCOUNTER >::operator!= ( const Pointer<
           T1, R1 > & right ) const [inline]
```

```
6.514.4.5  template<typename T, typename REFCOUNTER =
           AtomicRefCount> ReferenceType decaf::lang::Pointer< T,
           REFCOUNTER >::operator* ( ) const [inline]
```

```
6.514.4.6  template<typename T, typename REFCOUNTER =
           AtomicRefCount> ReferenceType decaf::lang::Pointer< T,
           REFCOUNTER >::operator* ( ) [inline]
```

Dereference Operator, returns a reference to the Contained value.

This method throws an `NullPointerException` if the contained value is `NULL`.

Returns

reference to the contained pointer.

Exceptions

NullPointerException if the contained value is `Null`

```
6.514.4.7  template<typename T, typename REFCOUNTER =  
           AtomicRefCount> PointerType decaf::lang::Pointer< T,  
           REFCOUNTER >::operator-> (    ) [inline]
```

Indirection Operator, returns a pointer to the Contained value.

This method throws an `NullPointerException` if the contained value is `NULL`.

Returns

reference to the contained pointer.

Exceptions

NullPointerException if the contained value is `Null`

```
6.514.4.8  template<typename T, typename REFCOUNTER =  
           AtomicRefCount> PointerType decaf::lang::Pointer< T,  
           REFCOUNTER >::operator-> (    ) const [inline]
```

```
6.514.4.9  template<typename T, typename REFCOUNTER =  
           AtomicRefCount> Pointer& decaf::lang::Pointer< T, REFCOUNTER  
           >::operator= ( const Pointer< T, REFCOUNTER > & right ) throw ()  
           [inline]
```

Assigns the value of *right* to this **Pointer** (p. 2343) and increments the reference Count.

Parameters

right - **Pointer** (p. 2343) on the right hand side of an operator= call to this.

6.514.4.10 `template<typename T, typename REFCOUNTER = AtomicRefCounter> template<typename T1, typename R1 > Pointer& decaf::lang::Pointer< T, REFCOUNTER >::operator= (const Pointer< T1, R1 > & right) throw () [inline]`

6.514.4.11 `template<typename T, typename REFCOUNTER = AtomicRefCounter> template<typename T1, typename R1 > bool decaf::lang::Pointer< T, REFCOUNTER >::operator== (const Pointer< T1, R1 > & right) const [inline]`

6.514.4.12 `template<typename T, typename REFCOUNTER = AtomicRefCounter> T* decaf::lang::Pointer< T, REFCOUNTER >::release () [inline]`

Releases the **Pointer** (p.2343) held and resets the internal pointer value to Null.

This method is not guaranteed to be safe if the **Pointer** (p. 2343) is held by more than one object or this method is called from more than one thread.

Parameters

value - The new value to contain.

Returns

The pointer instance that was held by this **Pointer** (p. 2343) object, the pointer is no longer owned by this **Pointer** (p. 2343) and won't be freed when this **Pointer** (p. 2343) goes out of scope.

Referenced by `decaf::lang::Pointer< TransactionId >::Pointer()`.

6.514.4.13 `template<typename T, typename REFCOUNTER = AtomicRefCounter> void decaf::lang::Pointer< T, REFCOUNTER >::reset (T * value) [inline]`

Resets the **Pointer** (p. 2343) to hold the new value.

Before the new value is stored reset checks if the old value should be destroyed and if so calls delete. Call reset with a value of NULL is supported and acts to set this **Pointer** (p. 2343) to a NULL pointer.

Parameters

value - The new value to contain.

6.514.4.14 `template<typename T, typename REFCOUNTER = AtomicRefCounter> template<typename T1 > Pointer<T1, CounterType> decaf::lang::Pointer< T, REFCOUNTER >::staticCast () const [inline]`

6.514.4.15 `template<typename T, typename REFCOUNTER = AtomicRefCounter> void decaf::lang::Pointer< T, REFCOUNTER >::swap (Pointer< T, REFCOUNTER > & value) throw () [inline]`

Exception (p. 1476) Safe Swap Function.

Parameters

value - the value to swap with this.

Referenced by decaf::lang::Pointer< TransactionId >::operator=(), and decaf::lang::Pointer< TransactionId >::swap().

6.514.5 Friends And Related Function Documentation

6.514.5.1 `template<typename T, typename REFCOUNTER = AtomicRefCount> bool operator!= (const Pointer< T, REFCOUNTER > & left, const T * right) [friend]`

6.514.5.2 `template<typename T, typename REFCOUNTER = AtomicRefCount> bool operator!= (const T * left, const Pointer< T, REFCOUNTER > & right) [friend]`

6.514.5.3 `template<typename T, typename REFCOUNTER = AtomicRefCount> bool operator== (const Pointer< T, REFCOUNTER > & left, const T * right) [friend]`

6.514.5.4 `template<typename T, typename REFCOUNTER = AtomicRefCount> bool operator== (const T * left, const Pointer< T, REFCOUNTER > & right) [friend]`

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Pointer.h`

6.515 decaf::lang::PointerComparator< T, R > Class Template Reference

This implementation of Comparator is designed to allows objects in a Collection to be sorted or tested for equality based on the value of the Object being Pointed to and not the value of the contained pointer in the **Pointer** (p. 2343) instance.

```
#include <src/main/decaf/lang/Pointer.h>
```

Inheritance diagram for decaf::lang::PointerComparator< T, R >:

Public Member Functions

- virtual bool **operator()** (const **Pointer**< T, R > &left, const **Pointer**< T, R > &right) const
- virtual int **compare** (const **Pointer**< T, R > &left, const **Pointer**< T, R > &right) const

6.515.1 Detailed Description

```
template<typename T, typename R = AtomicRefCounter> class
decaf::lang::PointerComparator< T, R >
```

This implementation of Comparator is designed to allows objects in a Collection to be sorted or tested for equality based on the value of the Object being Pointed to and not the value of the contained pointer in the **Pointer** (p. 2343) instance. This can be useful in the case where a series of values in a Collection is more efficiently accessed in the Objects Natural Order and not the underlying pointers location in memory.

Also this allows **Pointer** (p. 2343) objects that Point to different instances of the same type to be compared based on the comparison of the object itself and not just the value of the pointer.

6.515.2 Member Function Documentation

6.515.2.1 `template<typename T , typename R = AtomicRefCounter> virtual int
decaf::lang::PointerComparator< T, R >::compare (const Pointer< T,
R > & left, const Pointer< T, R > & right) const [inline, virtual]`

6.515.2.2 `template<typename T , typename R = AtomicRefCounter> virtual bool
decaf::lang::PointerComparator< T, R >::operator() (const Pointer< T,
R > & left, const Pointer< T, R > & right) const [inline, virtual]`

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Pointer.h`

6.516 activemq::cmsutil::PooledSession Class Reference

A pooled session object that wraps around a delegate session.

```
#include <src/main/activemq/cmsutil/PooledSession.h>
```

Inheritance diagram for `activemq::cmsutil::PooledSession`:

Public Member Functions

- **PooledSession** (**SessionPool** *pool, **cms::Session** *session)
- virtual **~PooledSession** ()
Does nothing.
- virtual **cms::Session** * **getSession** ()
Returns a non-constant reference to the internal session object.
- virtual const **cms::Session** * **getSession** () const
Returns a constant reference to the internal session object.
- virtual void **close** () throw (cms::CMSException)

Returns this session back to the pool, but does not close or destroy the internal session object.

- virtual void **commit** () throw (cms::CMSEException)
Commits all messages done in this transaction and releases any locks currently held.
- virtual void **rollback** () throw (cms::CMSEException)
Rolls back all messages done in this transaction and releases any locks currently held.
- virtual void **recover** () throw (cms::CMSEException)
Stops message delivery in this session, and restarts message delivery with the oldest unacknowledged message.
- virtual cms::MessageConsumer * **createConsumer** (const cms::Destination *destination) throw (cms::CMSEException)
Creates a MessageConsumer for the specified destination.
- virtual cms::MessageConsumer * **createConsumer** (const cms::Destination *destination, const std::string &selector) throw (cms::CMSEException)
Creates a MessageConsumer for the specified destination, using a message selector.
- virtual cms::MessageConsumer * **createConsumer** (const cms::Destination *destination, const std::string &selector, bool noLocal) throw (cms::CMSEException)
Creates a MessageConsumer for the specified destination, using a message selector.
- virtual cms::MessageConsumer * **createDurableConsumer** (const cms::Topic *destination, const std::string &name, const std::string &selector, bool noLocal=false) throw (cms::CMSEException)
Creates a durable subscriber to the specified topic, using a Message selector.
- virtual cms::MessageConsumer * **createCachedConsumer** (const cms::Destination *destination, const std::string &selector, bool noLocal) throw (cms::CMSEException)
First checks the internal consumer cache and creates one if none exist for the given destination, selector, noLocal.
- virtual cms::MessageProducer * **createProducer** (const cms::Destination *destination) throw (cms::CMSEException)
Creates a MessageProducer to send messages to the specified destination.
- virtual cms::MessageProducer * **createCachedProducer** (const cms::Destination *destination) throw (cms::CMSEException)
First checks the internal producer cache and creates one if none exist for the given destination.
- virtual cms::QueueBrowser * **createBrowser** (const cms::Queue *queue) throw (cms::CMSEException)
Creates a new QueueBrowser to peek at Messages on the given Queue.
- virtual cms::QueueBrowser * **createBrowser** (const cms::Queue *queue, const std::string &selector) throw (cms::CMSEException)
Creates a new QueueBrowser to peek at Messages on the given Queue.

- virtual **cms::Queue** * **createQueue** (const std::string &queueName) throw (cms::CMSEException)
Creates a queue identity given a Queue name.
- virtual **cms::Topic** * **createTopic** (const std::string &topicName) throw (cms::CMSEException)
Creates a topic identity given a Queue name.
- virtual **cms::TemporaryQueue** * **createTemporaryQueue** () throw (cms::CMSEException)
Creates a TemporaryQueue object.
- virtual **cms::TemporaryTopic** * **createTemporaryTopic** () throw (cms::CMSEException)
Creates a TemporaryTopic object.
- virtual **cms::Message** * **createMessage** () throw (cms::CMSEException)
Creates a new Message.
- virtual **cms::BytesMessage** * **createBytesMessage** () throw (cms::CMSEException)
Creates a BytesMessage.
- virtual **cms::BytesMessage** * **createBytesMessage** (const unsigned char *bytes, std::size_t bytesSize) throw (cms::CMSEException)
Creates a BytesMessage and sets the payload to the passed value.
- virtual **cms::StreamMessage** * **createStreamMessage** () throw (cms::CMSEException)
Creates a new StreamMessage.
- virtual **cms::TextMessage** * **createTextMessage** () throw (cms::CMSEException)
Creates a new TextMessage.
- virtual **cms::TextMessage** * **createTextMessage** (const std::string &text) throw (cms::CMSEException)
Creates a new TextMessage and set the text to the value given.
- virtual **cms::MapMessage** * **createMapMessage** () throw (cms::CMSEException)
Creates a new MapMessage.
- virtual **cms::Session::AcknowledgeMode** **getAcknowledgeMode** () const throw (cms::CMSEException)
Returns the acknowledgment mode of the session.
- virtual bool **isTransacted** () const throw (cms::CMSEException)
Gets if the Sessions is a Transacted Session.
- virtual void **unsubscribe** (const std::string &name) throw (cms::CMSEException)
Unsubscribes a durable subscription that has been created by a client.

6.516.1 Detailed Description

A pooled session object that wraps around a delegate session. Calls to close this session only result in giving the session back to the pool.

6.516.2 Constructor & Destructor Documentation

6.516.2.1 `activemq::cmsutil::PooledSession::PooledSession (SessionPool * pool, cms::Session * session)`

6.516.2.2 `virtual activemq::cmsutil::PooledSession::~~PooledSession ()` [virtual]

Does nothing.

6.516.3 Member Function Documentation

6.516.3.1 `virtual void activemq::cmsutil::PooledSession::close () throw (cms::CMSEException)` [virtual]

Returns this session back to the pool, but does not close or destroy the internal session object. Implements `cms::Session` (p. 2667).

6.516.3.2 `virtual void activemq::cmsutil::PooledSession::commit () throw (cms::CMSEException)` [inline, virtual]

Commits all messages done in this transaction and releases any locks currently held.

Exceptions

CMSEException - If an internal error occurs.

IllegalStateException - if the method is not called by a transacted session.

Implements `cms::Session` (p. 2667).

References `cms::Session::commit()`.

6.516.3.3 `virtual cms::QueueBrowser* activemq::cmsutil::PooledSession::createBrowser (const cms::Queue * queue) throw (cms::CMSEException)` [virtual]

Creates a new QueueBrowser to peek at Messages on the given Queue.

Parameters

queue the Queue to browse

Returns

New QueueBrowser that is owned by the caller.

Exceptions

CMSEException - If an internal error occurs.

InvalidDestinationException - if the destination given is invalid.

Implements **cms::Session** (p. 2668).

6.516.3.4 `virtual cms::QueueBrowser* activemq::cmsutil::PooledSession::createBrowser (const cms::Queue * queue, const std::string & selector) throw (cms::CMSEException)` [virtual]

Creates a new QueueBrowser to peek at Messages on the given Queue.

Parameters

queue the Queue to browse

selector the Message selector to filter which messages are browsed.

Returns

New QueueBrowser that is owned by the caller.

Exceptions

CMSEException - If an internal error occurs.

InvalidDestinationException - if the destination given is invalid.

Implements **cms::Session** (p. 2668).

6.516.3.5 `virtual cms::BytesMessage* activemq::cmsutil::PooledSession::createBytesMessage () throw (cms::CMSEException)` [inline, virtual]

Creates a BytesMessage.

Exceptions

CMSEException - If an internal error occurs.

Implements **cms::Session** (p. 2669).

References `cms::Session::createBytesMessage()`.

6.516.3.6 `virtual cms::BytesMessage* activemq::cmsutil::PooledSession::createBytesMessage (const unsigned char * bytes, std::size_t bytesSize) throw (cms::CMSEException)` [inline, virtual]

Creates a BytesMessage and sets the payload to the passed value.

Parameters

bytes an array of bytes to set in the message

bytesSize the size of the bytes array, or number of bytes to use

Exceptions

CMSEException - If an internal error occurs.

Implements **cms::Session** (p. 2669).

References **cms::Session::createBytesMessage()**.

6.516.3.7 `virtual cms::MessageConsumer* activemq::cmsutil::PooledSession::createCachedConsumer (const cms::Destination * destination, const std::string & selector, bool noLocal) throw (cms::CMSEException) [virtual]`

First checks the internal consumer cache and creates one if none exist for the given destination, selector, noLocal.

If created, the consumer is added to the pool's lifecycle manager.

Parameters

destination the destination to receive on

selector the selector to use

noLocal whether or not to receive messages from the same connection

Returns

the consumer resource

Exceptions

cms::CMSEException (p. 960) if something goes wrong.

6.516.3.8 `virtual cms::MessageProducer* activemq::cmsutil::PooledSession::createCachedProducer (const cms::Destination * destination) throw (cms::CMSEException) [virtual]`

First checks the internal producer cache and creates one if none exist for the given destination.

If created, the producer is added to the pool's lifecycle manager.

Parameters

destination the destination to send on

Returns

the producer resource

Exceptions

cms::CMSEException (p. 960) if something goes wrong.

6.516.3.9 `virtual cms::MessageConsumer* activemq::cmsutil::PooledSession::createConsumer (const cms::Destination * destination, const std::string & selector) throw (cms::CMSException)` [inline, virtual]

Creates a MessageConsumer for the specified destination, using a message selector.

Parameters

destination the Destination that this consumer receiving messages for.

selector the Message Selector to use

Returns

pointer to a new MessageConsumer that is owned by the caller (caller deletes)

Exceptions

CMSException - If an internal error occurs.

InvalidDestinationException - if an invalid destination is specified.

InvalidSelectorException - if the message selector is invalid.

Implements **cms::Session** (p. 2670).

References `cms::Session::createConsumer()`.

6.516.3.10 `virtual cms::MessageConsumer* activemq::cmsutil::PooledSession::createConsumer (const cms::Destination * destination, const std::string & selector, bool noLocal) throw (cms::CMSException)` [inline, virtual]

Creates a MessageConsumer for the specified destination, using a message selector.

Parameters

destination the Destination that this consumer receiving messages for.

selector the Message Selector to use

noLocal if true, and the destination is a topic, inhibits the delivery of messages published by its own connection. The behavior for NoLocal is not specified if the destination is a queue.

Returns

pointer to a new MessageConsumer that is owned by the caller (caller deletes)

Exceptions

CMSException - If an internal error occurs.

InvalidDestinationException - if an invalid destination is specified.

InvalidSelectorException - if the message selector is invalid.

Implements **cms::Session** (p. 2670).

References `cms::Session::createConsumer()`.


```

6.516.3.11 virtual cms::MessageConsumer* ac-
    tivemq::cmsutil::PooledSession::createConsumer ( const
    cms::Destination * destination ) throw ( cms::CMSException )
    [inline, virtual]

```

Creates a MessageConsumer for the specified destination.

Parameters

destination the Destination that this consumer receiving messages for.

Returns

pointer to a new MessageConsumer that is owned by the caller (caller deletes)

Exceptions

CMSException - If an internal error occurs.

InvalidDestinationException - if an invalid destination is specified.

Implements **cms::Session** (p. 2669).

References **cms::Session::createConsumer()**.

```

6.516.3.12 virtual cms::MessageConsumer* ac-
    tivemq::cmsutil::PooledSession::createDurableConsumer (
    const cms::Topic * destination, const std::string & name,
    const std::string & selector, bool noLocal = false ) throw (
    cms::CMSException ) [inline, virtual]

```

Creates a durable subscriber to the specified topic, using a Message selector.

Sessions that create durable consumers must use the same client Id as was used the last time the subscription was created in order to receive all messages that were delivered while the client was offline.

Parameters

destination the topic to subscribe to

name The name used to identify the subscription

selector the Message Selector to use

noLocal if true, and the destination is a topic, inhibits the delivery of messages published by its own connection. The behavior for NoLocal is not specified if the destination is a queue.

Returns

pointer to a new durable MessageConsumer that is owned by the caller (caller deletes)

Exceptions

CMSException - If an internal error occurs.

InvalidDestinationException - if an invalid destination is specified.

InvalidSelectorException - if the message selector is invalid.

Implements **cms::Session** (p. 2671).

References **cms::Session::createDurableConsumer()**.

6.516.3.13 `virtual cms::MapMessage* activemq::cmsutil::PooledSession::createMapMessage ()
throw (cms::CMSException) [inline, virtual]`

Creates a new MapMessage.

Exceptions

CMSException - If an internal error occurs.

Implements **cms::Session** (p. 2671).

References cms::Session::createMapMessage().

6.516.3.14 `virtual cms::Message* activemq::cmsutil::PooledSession::createMessage
() throw (cms::CMSException) [inline, virtual]`

Creates a new Message.

Exceptions

CMSException - If an internal error occurs.

Implements **cms::Session** (p. 2671).

References cms::Session::createMessage().

6.516.3.15 `virtual cms::MessageProducer* activemq::cmsutil::PooledSession::createProducer (const
cms::Destination * destination) throw (cms::CMSException)
[inline, virtual]`

Creates a MessageProducer to send messages to the specified destination.

Parameters

destination the Destination to send on

Returns

New MessageProducer that is owned by the caller.

Exceptions

CMSException - If an internal error occurs.

InvalidDestinationException - if an invalid destination is specified.

Implements **cms::Session** (p. 2672).

References cms::Session::createProducer().

6.516.3.16 `virtual cms::Queue* activemq::cmsutil::PooledSession::createQueue (const std::string & queueName) throw (cms::CMSException)
[inline, virtual]`

Creates a queue identity given a Queue name.

Parameters

queueName the name of the new Queue

Returns

new Queue pointer that is owned by the caller.

Exceptions

CMSEException - If an internal error occurs.

Implements **cms::Session** (p. 2672).

References cms::Session::createQueue().

6.516.3.17 `virtual cms::StreamMessage* activemq::cmsutil::PooledSession::createStreamMessage () throw (cms::CMSEException) [inline, virtual]`

Creates a new StreamMessage.

Exceptions

CMSEException - If an internal error occurs.

Implements **cms::Session** (p. 2672).

References cms::Session::createStreamMessage().

6.516.3.18 `virtual cms::TemporaryQueue* activemq::cmsutil::PooledSession::createTemporaryQueue () throw (cms::CMSEException) [inline, virtual]`

Creates a TemporaryQueue object.

Returns

new TemporaryQueue pointer that is owned by the caller.

Exceptions

CMSEException - If an internal error occurs.

Implements **cms::Session** (p. 2673).

References cms::Session::createTemporaryQueue().

6.516.3.19 `virtual cms::TemporaryTopic* activemq::cmsutil::PooledSession::createTemporaryTopic () throw (cms::CMSEException) [inline, virtual]`

Creates a TemporaryTopic object.

Exceptions

CMSEException - If an internal error occurs.

Implements **cms::Session** (p. 2673).

References cms::Session::createTemporaryTopic().

6.516.3.20 `virtual cms::TextMessage* activemq::cmsutil::PooledSession::createTextMessage ()
throw (cms::CMSException) [inline, virtual]`

Creates a new TextMessage.

Exceptions

CMSException - If an internal error occurs.

Implements **cms::Session** (p. 2674).

References cms::Session::createTextMessage().

6.516.3.21 `virtual cms::TextMessage* activemq::cmsutil::PooledSession::createTextMessage (const std::string & text) throw (cms::CMSException) [inline, virtual]`

Creates a new TextMessage and set the text to the value given.

Parameters

text the initial text for the message

Exceptions

CMSException - If an internal error occurs.

Implements **cms::Session** (p. 2673).

References cms::Session::createTextMessage().

6.516.3.22 `virtual cms::Topic* activemq::cmsutil::PooledSession::createTopic (const std::string & topicName) throw (cms::CMSException) [inline, virtual]`

Creates a topic identity given a Queue name.

Parameters

topicName the name of the new Topic

Returns

new Topic pointer that is owned by the caller.

Exceptions

CMSException - If an internal error occurs.

Implements **cms::Session** (p. 2674).

References cms::Session::createTopic().

6.516.3.23 `virtual cms::Session::AcknowledgeMode activemq::cmsutil::PooledSession::getAcknowledgeMode () const throw (cms::CMSEException) [inline, virtual]`

Returns the acknowledgment mode of the session.

Returns

the Sessions Acknowledge Mode

Exceptions

CMSEException - If an internal error occurs.

Implements **cms::Session** (p. 2674).

References cms::Session::getAcknowledgeMode().

6.516.3.24 `virtual const cms::Session* activemq::cmsutil::PooledSession::getSession () const [inline, virtual]`

Returns a constant reference to the internal session object.

Returns

the session object.

6.516.3.25 `virtual cms::Session* activemq::cmsutil::PooledSession::getSession () [inline, virtual]`

Returns a non-constant reference to the internal session object.

Returns

the session object.

6.516.3.26 `virtual bool activemq::cmsutil::PooledSession::isTransacted () const throw (cms::CMSEException) [inline, virtual]`

Gets if the Sessions is a Transacted Session.

Returns

transacted true - false.

Exceptions

CMSEException - If an internal error occurs.

Implements **cms::Session** (p. 2674).

References cms::Session::isTransacted().

6.516.3.27 `virtual void activemq::cmsutil::PooledSession::recover () throw (cms::CMSException) [inline, virtual]`

Stops message delivery in this session, and restarts message delivery with the oldest unacknowledged message.

All consumers deliver messages in a serial order. Acknowledging a received message automatically acknowledges all messages that have been delivered to the client.

Restarting a session causes it to take the following actions:

- Stop message delivery
- Mark all messages that might have been delivered but not acknowledged as "redelivered"
- Restart the delivery sequence including all unacknowledged messages that had been previously delivered. Redelivered messages do not have to be delivered in exactly their original delivery order.

Exceptions

CMSException - if the CMS provider fails to stop and restart message delivery due to some internal error.

IllegalStateException - if the method is called by a transacted session.

Implements **cms::Session** (p. 2675).

References `cms::Session::recover()`.

6.516.3.28 `virtual void activemq::cmsutil::PooledSession::rollback () throw (cms::CMSException) [inline, virtual]`

Rolls back all messages done in this transaction and releases any locks currently held.

Exceptions

CMSException - If an internal error occurs.

IllegalStateException - if the method is not called by a transacted session.

Implements **cms::Session** (p. 2675).

References `cms::Session::rollback()`.

6.516.3.29 `virtual void activemq::cmsutil::PooledSession::unsubscribe (const std::string & name) throw (cms::CMSException) [inline, virtual]`

Unsubscribes a durable subscription that has been created by a client.

This method deletes the state being maintained on behalf of the subscriber by its provider. It is erroneous for a client to delete a durable subscription while there is an active MessageConsumer or Subscriber for the subscription, or while a consumed message is part of a pending transaction or has not been acknowledged in the session.

Parameters

name The name used to identify this subscription

Exceptions

CMSEException - If an internal error occurs.

Implements **cms::Session** (p. 2676).

References cms::Session::unsubscribe().

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/**PooledSession.h**

6.517 decaf::util::concurrent::PooledThread Class Reference

```
#include <src/main/decaf/util/concurrent/PooledThread.h>
```

Public Member Functions

- **PooledThread** (**ThreadPool** *pool)
Constructor.
- virtual **~PooledThread** ()
- virtual void **run** ()
*Run Method for this object waits for something to be enqueued on the **ThreadPool** (p. 2977) and then grabs it and calls its run method.*
- virtual void **stop** () throw (lang::Exception)
Stops the Thread, thread will complete its task if currently running one, and then die.
- virtual bool **isBusy** ()
Checks to see if the thread is busy, if busy it means that this thread has taken a task from the ThreadPool's queue and is processing it.
- virtual void **setPooledThreadListener** (**PooledThreadListener** *listener)
*Adds a listener to this **PooledThread** (p. 2364) to be notified when this thread starts and completes a task.*
- virtual **PooledThreadListener** * **getPooledThreadListener** ()
*Removes a listener for this **PooledThread** (p. 2364) to be notified when this thread starts and completes a task.*

6.517.1 Constructor & Destructor Documentation

6.517.1.1 decaf::util::concurrent::PooledThread::PooledThread (**ThreadPool** * pool)

Constructor.

Parameters

pool the parant **ThreadPool** (p. 2977) object

6.517.1.2 `virtual decaf::util::concurrent::PooledThread::~~PooledThread ()`
[virtual]

6.517.2 Member Function Documentation

6.517.2.1 `virtual PooledThreadListener* decaf::util::concurrent::PooledThread::getPooledThreadListener ()`
[inline, virtual]

Removes a listener for this `PooledThread` (p. 2364) to be notified when this thread starts and completes a task.

Returns

a pointer to this thread's listener or NULL

6.517.2.2 `virtual bool decaf::util::concurrent::PooledThread::isBusy ()` [inline, virtual]

Checks to see if the thread is busy, if busy it means that this thread has taken a task from the `ThreadPool`'s queue and is processing it.

Returns

true if the Thread is busy

6.517.2.3 `virtual void decaf::util::concurrent::PooledThread::run ()` [virtual]

Run Method for this object waits for something to be enqueued on the `ThreadPool` (p. 2977) and then grabs it and calls its run method.

6.517.2.4 `virtual void decaf::util::concurrent::PooledThread::setPooledThreadListener (PooledThreadListener * listener)` [inline, virtual]

Adds a listener to this `PooledThread` (p. 2364) to be notified when this thread starts and completes a task.

Parameters

listener the listener to send notifications to.

6.517.2.5 `virtual void decaf::util::concurrent::PooledThread::stop () throw (lang::Exception)` [virtual]

Stops the Thread, thread will complete its task if currently running one, and then die.

Does not block.

Exceptions

Exception

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/PooledThread.h`

6.518 decaf::util::concurrent::PooledThreadListener Class Reference

Abstract Listener Interface for users of `ThreadPool` (p. 2977).

```
#include <src/main/decaf/util/concurrent/PooledThreadListener.h>
```

Inheritance diagram for `decaf::util::concurrent::PooledThreadListener`:

Public Member Functions

- virtual `~PooledThreadListener ()`
- virtual void `onTaskStarted (PooledThread *thread)=0`

Called by a pooled thread when it is about to begin executing a new task.

- virtual void `onTaskCompleted (PooledThread *thread)=0`

Called by a pooled thread when it has completed a task and is going back to waiting for another task to run.

- virtual void `onTaskException (PooledThread *thread, lang::Exception &ex)=0`

*Called by a pooled thread when it has encountered an exception while running a user task, after receiving this notification the callee should assume that the **PooledThread** (p. 2364) is now no longer running.*

6.518.1 Detailed Description

Abstract Listener Interface for users of `ThreadPool` (p. 2977). The implementor of this class receives events related to the execution and termination of threads running in the **ThreadPool** (p. 2977).

Since

1.0

6.518.2 Constructor & Destructor Documentation

6.518.2.1 virtual
`decaf::util::concurrent::PooledThreadListener::~~PooledThreadListener (`
`) [inline, virtual]`

6.518.3 Member Function Documentation

6.518.3.1 virtual void `de-`
`caf::util::concurrent::PooledThreadListener::onTaskCompleted (`
`PooledThread * thread) [pure virtual]`

Called by a pooled thread when it has completed a task and is going back to waiting for another task to run.

Parameters

thread - Pointer the the Pooled Thread that is making this call.

Implemented in `decaf::util::concurrent::ThreadPool` (p. 2981).

6.518.3.2 virtual void `de-`
`caf::util::concurrent::PooledThreadListener::onTaskException (`
`PooledThread * thread, lang::Exception & ex) [pure virtual]`

Called by a pooled thread when it has encountered an exception while running a user task, after receiving this notification the callee should assume that the **PooledThread** (p. 2364) is now no longer running.

Parameters

thread - Pointer to the Pooled Thread that is making this call

ex - The Exception that occurred.

Implemented in `decaf::util::concurrent::ThreadPool` (p. 2981).

6.518.3.3 virtual void `decaf::util::concurrent::PooledThreadListener::onTaskStarted`
`(PooledThread * thread) [pure virtual]`

Called by a pooled thread when it is about to begin executing a new task.

Parameters

thread - Pointer to the Pooled Thread that is making this call

Implemented in `decaf::util::concurrent::ThreadPool` (p. 2981).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/PooledThreadListener.h`

6.519 decaf::net::PortUnreachableException Class Reference

```
#include <src/main/decaf/net/PortUnreachableException.h>
```

Inheritance diagram for decaf::net::PortUnreachableException:

Public Member Functions

- **PortUnreachableException** () throw ()
Default Constructor.
- **PortUnreachableException** (const Exception &ex) throw ()
Conversion Constructor from some other Exception.
- **PortUnreachableException** (const **PortUnreachableException** &ex) throw ()
Copy Constructor.
- **PortUnreachableException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **PortUnreachableException** (const std::exception *cause) throw ()
Constructor.
- **PortUnreachableException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **PortUnreachableException** * clone () const
Clones this exception.
- virtual ~**PortUnreachableException** () throw ()

6.519.1 Constructor & Destructor Documentation

6.519.1.1 decaf::net::PortUnreachableException::PortUnreachableException () throw () [inline]

Default Constructor.

6.519.1.2 decaf::net::PortUnreachableException::PortUnreachableException (const Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

Parameters

ex An exception that should become this type of Exception

6.519.1.3 `decaf::net::PortUnreachableException::PortUnreachableException (const PortUnreachableException & ex) throw () [inline]`

Copy Constructor.

Parameters

ex An exception that should become this type of Exception

6.519.1.4 `decaf::net::PortUnreachableException::PortUnreachableException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.519.1.5 `decaf::net::PortUnreachableException::PortUnreachableException (const std::exception * cause) throw () [inline]`

Constructor.

Parameters

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.519.1.6 `decaf::net::PortUnreachableException::PortUnreachableException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.519.1.7 `virtual`
`decaf::net::PortUnreachableException::~~PortUnreachableException ()`
`throw ()` [`inline`, `virtual`]

6.519.2 Member Function Documentation

6.519.2.1 `virtual PortUnreachableException* de-`
`caf::net::PortUnreachableException::clone () const`
`[inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

a new Exception instance that is a copy of this Exception object.

Reimplemented from `decaf::net::SocketException` (p. 2794).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/PortUnreachableException.h`

6.520 activemq::util::PrimitiveList Class Reference

List of primitives.

```
#include <src/main/activemq/util/PrimitiveList.h>
```

Inheritance diagram for `activemq::util::PrimitiveList`:

Public Member Functions

- `PrimitiveList ()`
Default Constructor, creates an Empty list.
- `virtual ~PrimitiveList ()`
- `PrimitiveList (const decaf::util::List< PrimitiveValueNode > &src)`
Copy Constructor.
- `PrimitiveList (const PrimitiveList &src)`
Copy Constructor.
- `std::string toString () const`
Converts the contents into a formatted string that can be output in a Log File or other debugging tool.

- virtual bool **getBool** (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException)

Gets the Boolean value at the specified index.

- virtual void **setBool** (std::size_t index, bool value) throw (decaf::lang::exceptions::IndexOutOfBoundsException)

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

- virtual unsigned char **getByte** (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException)

Gets the Byte value at the specified index.

- virtual void **setByte** (std::size_t index, unsigned char value) throw (decaf::lang::exceptions::IndexOutOfBoundsException)

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

- virtual char **getChar** (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException)

Gets the Character value at the specified index.

- virtual void **setChar** (std::size_t index, char value) throw (decaf::lang::exceptions::IndexOutOfBoundsException)

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

- virtual short **getShort** (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException)

Gets the Short value at the specified index.

- virtual void **setShort** (std::size_t index, short value) throw (decaf::lang::exceptions::IndexOutOfBoundsException)

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

- virtual int **getInt** (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException)

Gets the Integer value at the specified index.

- virtual void **setInt** (std::size_t index, int value) throw (decaf::lang::exceptions::IndexOutOfBoundsException)

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

- virtual long long **getLong** (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException)

Gets the Long value at the specified index.

- virtual void **setLong** (std::size_t index, long long value) throw (decaf::lang::exceptions::IndexOutOfBoundsException)

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

- virtual float **getFloat** (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException)

Gets the Float value at the specified index.

- virtual void **setFloat** (std::size_t index, float value) throw (decaf::lang::exceptions::IndexOutOfBoundsException)

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

- virtual double **getDouble** (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException)

Gets the Double value at the specified index.

- virtual void **setDouble** (std::size_t index, double value) throw (decaf::lang::exceptions::IndexOutOfBoundsException)

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

- virtual std::string **getString** (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException)

Gets the String value at the specified index.

- virtual void **setString** (std::size_t index, const std::string &value) throw (decaf::lang::exceptions::IndexOutOfBoundsException)

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

- virtual std::vector< unsigned char > **getByteArray** (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException)

Gets the Byte Array value at the specified index.

- virtual void **setByteArray** (std::size_t index, const std::vector< unsigned char > &value) throw (decaf::lang::exceptions::IndexOutOfBoundsException)

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

6.520.1 Detailed Description

List of primitives.

6.520.2 Constructor & Destructor Documentation

6.520.2.1 activemq::util::PrimitiveList::PrimitiveList ()

Default Constructor, creates an Empty list.

6.520.2.2 virtual activemq::util::PrimitiveList::~~PrimitiveList () [virtual]

6.520.2.3 activemq::util::PrimitiveList::PrimitiveList (const decaf::util::List< PrimitiveValueNode > & src)

Copy Constructor.

Parameters

src - the Decaf List of PrimitiveNodeValues to copy

6.520.2.4 activemq::util::PrimitiveList::PrimitiveList (const PrimitiveList & src)

Copy Constructor.

Parameters

src - the **PrimitiveList** (p. 2370) to copy

6.520.3 Member Function Documentation

6.520.3.1 virtual bool activemq::util::PrimitiveList::getBool (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]

Gets the Boolean value at the specified index.

Parameters

index - index to get value from

Returns

value contained at the given index

Exceptions

IndexOutOfBoundsException if index is > `size()` (p. 2844)

UnsupportedOperationException if the type at index is not of the type that this method is to return or can convert to.

6.520.3.2 virtual unsigned char activemq::util::PrimitiveList::getBytes (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]

Gets the Byte value at the specified index.

Parameters

index - index to get value from

Returns

value contained at the given index

Exceptions

IndexOutOfBoundsException if index is > `size()` (p. 2844)

UnsupportedOperationException if the type at index is not of the type that this method is to return or can convert to.

6.520.3.3 virtual std::vector<unsigned char> activemq::util::PrimitiveList::getBytesArray (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]

Gets the Byte Array value at the specified index.

Parameters

index - index to get value from

Returns

value contained at the given index

Exceptions

IndexOutOfBoundsException if index is > `size()` (p. 2844)

UnsupportedOperationException if the type at index is not of the type that this method is to return or can convert to.

6.520.3.4 `virtual char activemq::util::PrimitiveList::getChar (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Gets the Character value at the specified index.

Parameters

index - index to get value from

Returns

value contained at the given index

Exceptions

IndexOutOfBoundsException if index is > `size()` (p. 2844)

UnsupportedOperationException if the type at index is not of the type that this method is to return or can convert to.

6.520.3.5 `virtual double activemq::util::PrimitiveList::getDouble (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Gets the Double value at the specified index.

Parameters

index - index to get value from

Returns

value contained at the given index

Exceptions

IndexOutOfBoundsException if index is > `size()` (p. 2844)

UnsupportedOperationException if the type at index is not of the type that this method is to return or can convert to.

6.520.3.6 `virtual float activemq::util::PrimitiveList::getFloat (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Gets the Float value at the specified index.

Parameters

index - index to get value from

Returns

value contained at the given index

Exceptions

IndexOutOfBoundsException if index is > **size()** (p. 2844)

UnsupportedOperationException if the type at index is not of the type that this method is to return or can convert to.

6.520.3.7 `virtual int activemq::util::PrimitiveList::getInt (std::size_t index)
 const throw (decaf::lang::exceptions::IndexOutOfBoundsException,
 decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Gets the Integer value at the specified index.

Parameters

index - index to get value from

Returns

value contained at the given index

Exceptions

IndexOutOfBoundsException if index is > **size()** (p. 2844)

UnsupportedOperationException if the type at index is not of the type that this method is to return or can convert to.

6.520.3.8 `virtual long long activemq::util::PrimitiveList::getLong (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException,
 decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Gets the Long value at the specified index.

Parameters

index - index to get value from

Returns

value contained at the given index

Exceptions

IndexOutOfBoundsException if index is > **size()** (p. 2844)

UnsupportedOperationException if the type at index is not of the type that this method is to return or can convert to.

6.520.3.9 `virtual short activemq::util::PrimitiveList::getShort (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException,
 decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Gets the Short value at the specified index.

Parameters

index - index to get value from

Returns

value contained at the given index

Exceptions

IndexOutOfBoundsException if index is > `size()` (p. 2844)

UnsupportedOperationException if the type at index is not of the type that this method is to return or can convert to.

```
6.520.3.10  virtual std::string activemq::util::PrimitiveList::getString  
            ( std::size_t  index ) const throw ( de-  
            caf::lang::exceptions::IndexOutOfBoundsException,  
            decaf::lang::exceptions::UnsupportedOperationException ) [virtual]
```

Gets the String value at the specified index.

Parameters

index - index to get value from

Returns

value contained at the given index

Exceptions

IndexOutOfBoundsException if index is > `size()` (p. 2844)

UnsupportedOperationException if the type at index is not of the type that this method is to return or can convert to.

```
6.520.3.11  virtual void activemq::util::PrimitiveList::setBool  
            ( std::size_t  index,  bool  value ) throw ( decaf::lang::exceptions::IndexOutOfBoundsException ) [virtual]
```

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

Parameters

index - location to set in the list

value - the new value to assign to the element at index

Exceptions

IndexOutOfBoundsException if index > `size()` (p. 2844).

6.520.3.12 `virtual void activemq::util::PrimitiveList::setByte (`
 `std::size_t index, unsigned char value) throw (`
 `decaf::lang::exceptions::IndexOutOfBoundsException) [virtual]`

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

Parameters

index - location to set in the list

value - the new value to assign to the element at index

Exceptions

IndexOutOfBoundsException if `index > size()` (p. 2844).

6.520.3.13 `virtual void activemq::util::PrimitiveList::setByteArray (std::size_t`
 `index, const std::vector< unsigned char > & value) throw (`
 `decaf::lang::exceptions::IndexOutOfBoundsException) [virtual]`

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

Parameters

index - location to set in the list

value - the new value to assign to the element at index

Exceptions

IndexOutOfBoundsException if `index > size()` (p. 2844).

6.520.3.14 `virtual void activemq::util::PrimitiveList::setChar`
 `(std::size_t index, char value) throw (`
 `decaf::lang::exceptions::IndexOutOfBoundsException) [virtual]`

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

Parameters

index - location to set in the list

value - the new value to assign to the element at index

Exceptions

IndexOutOfBoundsException if `index > size()` (p. 2844).

6.520.3.15 `virtual void activemq::util::PrimitiveList::setDouble
(std::size_t index, double value) throw (decaf::lang::exceptions::IndexOutOfBoundsException) [virtual]`

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

Parameters

index - location to set in the list

value - the new value to assign to the element at index

Exceptions

IndexOutOfBoundsException if index > size() (p. 2844).

6.520.3.16 `virtual void activemq::util::PrimitiveList::setFloat
(std::size_t index, float value) throw (decaf::lang::exceptions::IndexOutOfBoundsException) [virtual]`

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

Parameters

index - location to set in the list

value - the new value to assign to the element at index

Exceptions

IndexOutOfBoundsException if index > size() (p. 2844).

6.520.3.17 `virtual void activemq::util::PrimitiveList::setInt (std::size_t index, int
value) throw (decaf::lang::exceptions::IndexOutOfBoundsException)
[virtual]`

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

Parameters

index - location to set in the list

value - the new value to assign to the element at index

Exceptions

IndexOutOfBoundsException if index > size() (p. 2844).

6.520.3.18 **virtual void activemq::util::PrimitiveList::setLong**
 (**std::size_t** *index*, **long long** *value*) **throw (**
 decaf::lang::exceptions::IndexOutOfBoundsException) [virtual]

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

Parameters

index - location to set in the list

value - the new value to assign to the element at index

Exceptions

IndexOutOfBoundsException if index > size() (p. 2844).

6.520.3.19 **virtual void activemq::util::PrimitiveList::setShort**
 (**std::size_t** *index*, **short** *value*) **throw (**
 decaf::lang::exceptions::IndexOutOfBoundsException) [virtual]

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

Parameters

index - location to set in the list

value - the new value to assign to the element at index

Exceptions

IndexOutOfBoundsException if index > size() (p. 2844).

6.520.3.20 **virtual void activemq::util::PrimitiveList::setString (**
 std::size_t *index*, **const std::string &** *value*) **throw (**
 decaf::lang::exceptions::IndexOutOfBoundsException) [virtual]

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

Parameters

index - location to set in the list

value - the new value to assign to the element at index

Exceptions

IndexOutOfBoundsException if index > size() (p. 2844).

6.520.3.21 `std::string activemq::util::PrimitiveList::toString () const`

Converts the contents into a formatted string that can be output in a Log File or other debugging tool.

Returns

formatted String of all elements in the list.

The documentation for this class was generated from the following file:

- `src/main/activemq/util/PrimitiveList.h`

6.521 `activemq::util::PrimitiveMap` Class Reference

Map of named primitives.

```
#include <src/main/activemq/util/PrimitiveMap.h>
```

Inheritance diagram for `activemq::util::PrimitiveMap`:

Public Member Functions

- **PrimitiveMap** ()
Default Constructor, creates an empty map.
- virtual **~PrimitiveMap** ()
- **PrimitiveMap** (const **decaf::util::Map**< `std::string`, **PrimitiveValueNode** > &source)
Copy Constructor.
- **PrimitiveMap** (const **PrimitiveMap** &source)
Copy Constructor.
- `std::string toString` () const
Converts the contents into a formatted string that can be output in a Log File or other debugging tool.
- virtual `bool getBool` (const `std::string` &key) const
throw (`decaf::lang::exceptions::NoSuchElementException`, `decaf::lang::exceptions::UnsupportedOperationException`)
Gets the Boolean value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type `NoSuchElementException` is thrown.
- virtual void **setBool** (const `std::string` &key, bool value)
Sets the value at key to the specified type.
- virtual `unsigned char getByte` (const `std::string` &key) const
throw (`decaf::lang::exceptions::NoSuchElementException`, `decaf::lang::exceptions::UnsupportedOperationException`)

Gets the Byte value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type `NoSuchElementException` is thrown.

- virtual void **setByte** (const std::string &key, unsigned char value)

Sets the value at key to the specified type.

- virtual char **getChar** (const std::string &key) const
throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException)

Gets the Character value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type `NoSuchElementException` is thrown.

- virtual void **setChar** (const std::string &key, char value)

Sets the value at key to the specified type.

- virtual short **getShort** (const std::string &key) const
throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException)

Gets the Short value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type `NoSuchElementException` is thrown.

- virtual void **setShort** (const std::string &key, short value)

Sets the value at key to the specified type.

- virtual int **getInt** (const std::string &key) const throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException)

Gets the Integer value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type `NoSuchElementException` is thrown.

- virtual void **setInt** (const std::string &key, int value)

Sets the value at key to the specified type.

- virtual long long **getLong** (const std::string &key) const
throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException)

Gets the Long value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type `NoSuchElementException` is thrown.

- virtual void **setLong** (const std::string &key, long long value)

Sets the value at key to the specified type.

- virtual float **getFloat** (const std::string &key) const
throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException)

Gets the Float value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type `NoSuchElementException` is thrown.

- virtual void **setFloat** (const std::string &key, float value)

Sets the value at key to the specified type.

- virtual double **getDouble** (const std::string &key) const
throw (decaf::lang::exceptions::NoSuchElementException, de-
caf::lang::exceptions::UnsupportedOperationException)

Gets the Double value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type `NoSuchElementException` is thrown.

- virtual void **setDouble** (const std::string &key, double value)

Sets the value at key to the specified type.

- virtual std::string **getString** (const std::string &key) const
throw (decaf::lang::exceptions::NoSuchElementException, de-
caf::lang::exceptions::UnsupportedOperationException)

Gets the String value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type `NoSuchElementException` is thrown.

- virtual void **setString** (const std::string &key, const std::string &value)

Sets the value at key to the specified type.

- virtual std::vector< unsigned char > **getByteArray** (const std::string &key) const
throw (decaf::lang::exceptions::NoSuchElementException, de-
caf::lang::exceptions::UnsupportedOperationException)

Gets the Byte Array value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type `NoSuchElementException` is thrown.

- virtual void **setByteArray** (const std::string &key, const std::vector< unsigned char > &value)

Sets the value at key to the specified type.

6.521.1 Detailed Description

Map of named primitives.

6.521.2 Constructor & Destructor Documentation

6.521.2.1 activemq::util::PrimitiveMap::PrimitiveMap ()

Default Constructor, creates an empty map.

6.521.2.2 virtual activemq::util::PrimitiveMap::~~PrimitiveMap () [virtual]

6.521.2.3 activemq::util::PrimitiveMap::PrimitiveMap (const decaf::util::Map< std::string, PrimitiveValueNode > & source)

Copy Constructor.

Parameters

source The Decaf Library Map instance whose elements will be copied into this Map.

6.521.2.4 `activemq::util::PrimitiveMap::PrimitiveMap (const PrimitiveMap & source)`

Copy Constructor.

Parameters

source The **PrimitiveMap** (p.2381) whose elements will be copied into this Map.

6.521.3 Member Function Documentation

6.521.3.1 `virtual bool activemq::util::PrimitiveMap::getBool (const std::string & key) const throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Gets the Boolean value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type `NoSuchElementException` is thrown.

Parameters

key - the location to return the value from.

Returns

the value at *key* in the type requested.

Exceptions

NoSuchElementException if key is not in the map.

UnsupportedOperationException if the value cannot be converted to the type this method returns

6.521.3.2 `virtual unsigned char activemq::util::PrimitiveMap::getBytes (const std::string & key) const throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Gets the Byte value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type `NoSuchElementException` is thrown.

Parameters

key - the location to return the value from.

Returns

the value at *key* in the type requested.

Exceptions

NoSuchElementException if key is not in the map.

UnsupportedOperationException if the value cannot be converted to the type this method returns

6.521.3.3 `virtual std::vector<unsigned char> activemq::util::PrimitiveMap::getByteArray (const std::string & key) const throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Gets the Byte Array value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.

Parameters

key - the location to return the value from.

Returns

the value at key in the type requested.

Exceptions

NoSuchElementException if key is not in the map.

UnsupportedOperationException if the value cannot be converted to the type this method returns

6.521.3.4 `virtual char activemq::util::PrimitiveMap::getChar (const std::string & key) const throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Gets the Character value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.

Parameters

key - the location to return the value from.

Returns

the value at key in the type requested.

Exceptions

NoSuchElementException if key is not in the map.

UnsupportedOperationException if the value cannot be converted to the type this method returns

6.521.3.5 `virtual double activemq::util::PrimitiveMap::getDouble (const std::string & key) const throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Gets the Double value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.

Parameters

key - the location to return the value from.

Returns

the value at key in the type requested.

Exceptions

NoSuchElementException if key is not in the map.

UnsupportedOperationException if the value cannot be converted to the type this method returns

6.521.3.6 `virtual float activemq::util::PrimitiveMap::getFloat (const std::string & key) const throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Gets the Float value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.

Parameters

key - the location to return the value from.

Returns

the value at key in the type requested.

Exceptions

NoSuchElementException if key is not in the map.

UnsupportedOperationException if the value cannot be converted to the type this method returns

6.521.3.7 `virtual int activemq::util::PrimitiveMap::getInt (const std::string & key) const throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Gets the Integer value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.

Parameters

key - the location to return the value from.

Returns

the value at key in the type requested.

Exceptions

NoSuchElementException if key is not in the map.

UnsupportedOperationException if the value cannot be converted to the type this method returns

6.521.3.8 `virtual long long activemq::util::PrimitiveMap::getLong (const std::string & key) const throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Gets the Long value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type `NoSuchElementException` is thrown.

Parameters

key - the location to return the value from.

Returns

the value at *key* in the type requested.

Exceptions

NoSuchElementException if key is not in the map.

UnsupportedOperationException if the value cannot be converted to the type this method returns

6.521.3.9 `virtual short activemq::util::PrimitiveMap::getShort (const std::string & key) const throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Gets the Short value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type `NoSuchElementException` is thrown.

Parameters

key - the location to return the value from.

Returns

the value at *key* in the type requested.

Exceptions

NoSuchElementException if key is not in the map.

UnsupportedOperationException if the value cannot be converted to the type this method returns

6.521.3.10 `virtual std::string activemq::util::PrimitiveMap::getString (const std::string & key) const throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Gets the String value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type `NoSuchElementException` is thrown.

Parameters

key - the location to return the value from.

Returns

the value at key in the type requested.

Exceptions

NoSuchElementException if key is not in the map.

UnsupportedOperationException if the value cannot be converted to the type this method returns

6.521.3.11 virtual void activemq::util::PrimitiveMap::setBool (const std::string & key, bool value) [virtual]

Sets the value at key to the specified type.

Overwrites any data that was previously at this key or inserts a new element at key.

Parameters

key - the map key to set or insert.

value - the new value to set at the key location.

6.521.3.12 virtual void activemq::util::PrimitiveMap::setByte (const std::string & key, unsigned char value) [virtual]

Sets the value at key to the specified type.

Overwrites any data that was previously at this key or inserts a new element at key.

Parameters

key - the map key to set or insert.

value - the new value to set at the key location.

6.521.3.13 virtual void activemq::util::PrimitiveMap::setByteArray (const std::string & key, const std::vector< unsigned char > & value) [virtual]

Sets the value at key to the specified type.

Overwrites any data that was previously at this key or inserts a new element at key.

Parameters

key - the map key to set or insert.

value - the new value to set at the key location.

6.521.3.14 virtual void activemq::util::PrimitiveMap::setChar (const std::string & key, char value) [virtual]

Sets the value at key to the specified type.

Overwrites any data that was previously at this key or inserts a new element at key.

Parameters

key - the map key to set or insert.

value - the new value to set at the key location.

6.521.3.15 `virtual void activemq::util::PrimitiveMap::setDouble (const std::string & key, double value) [virtual]`

Sets the value at key to the specified type.

Overwrites any data that was previously at this key or inserts a new element at key.

Parameters

key - the map key to set or insert.

value - the new value to set at the key location.

6.521.3.16 `virtual void activemq::util::PrimitiveMap::setFloat (const std::string & key, float value) [virtual]`

Sets the value at key to the specified type.

Overwrites any data that was previously at this key or inserts a new element at key.

Parameters

key - the map key to set or insert.

value - the new value to set at the key location.

6.521.3.17 `virtual void activemq::util::PrimitiveMap::setInt (const std::string & key, int value) [virtual]`

Sets the value at key to the specified type.

Overwrites any data that was previously at this key or inserts a new element at key.

Parameters

key - the map key to set or insert.

value - the new value to set at the key location.

6.521.3.18 `virtual void activemq::util::PrimitiveMap::setLong (const std::string & key, long long value) [virtual]`

Sets the value at key to the specified type.

Overwrites any data that was previously at this key or inserts a new element at key.

Parameters

key - the map key to set or insert.

value - the new value to set at the key location.

6.521.3.19 `virtual void activemq::util::PrimitiveMap::setShort (const std::string & key, short value)` [virtual]

Sets the value at key to the specified type.

Overwrites any data that was previously at this key or inserts a new element at key.

Parameters

key - the map key to set or insert.

value - the new value to set at the key location.

6.521.3.20 `virtual void activemq::util::PrimitiveMap::setString (const std::string & key, const std::string & value)` [virtual]

Sets the value at key to the specified type.

Overwrites any data that was previously at this key or inserts a new element at key.

Parameters

key - the map key to set or insert.

value - the new value to set at the key location.

6.521.3.21 `std::string activemq::util::PrimitiveMap::toString ()` const

Converts the contents into a formatted string that can be output in a Log File or other debugging tool.

Returns

formatted String of all elements in the map.

The documentation for this class was generated from the following file:

- `src/main/activemq/util/PrimitiveMap.h`

6.522 **activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller Class Reference**

This class wraps the functionality needed to marshal a primitive map to the Openwire Format's expectation of what the map looks like on the wire.

```
#include <src/main/activemq/wireformat/openwire/marshal/PrimitiveTypesMarshaller.h>
```

Public Member Functions

- `PrimitiveTypesMarshaller ()`
- `virtual ~PrimitiveTypesMarshaller ()`

Static Public Member Functions

- static void **marshal** (const **util::PrimitiveMap** *map, std::vector< unsigned char > &dest) throw (decaf::lang::Exception)
Static Marshal of a primitive map object.
- static void **unmarshal** (**util::PrimitiveMap** *map, const std::vector< unsigned char > &src) throw (decaf::lang::Exception)
Static Map Unmarshaler, takes an array of bytes and returns a new instance of a PrimitiveMap object.
- static void **marshal** (const **util::PrimitiveList** *list, std::vector< unsigned char > &dest) throw (decaf::lang::Exception)
Static Marshal of a primitive map object.
- static void **unmarshal** (**util::PrimitiveList** *list, const std::vector< unsigned char > &src) throw (decaf::lang::Exception)
Static Map Unmarshaler, takes an array of bytes and returns a new instance of a PrimitiveMap object.

Static Protected Member Functions

- static void **marshalPrimitiveMap** (decaf::io::DataOutputStream &dataOut, const decaf::util::Map< std::string, util::PrimitiveValueNode > &map) throw (decaf::io::IOException)
Marshal a Map of Primitives to the given OutputStream, can result in recursive calls to this method if the map contains maps of maps.
- static void **marshalPrimitiveList** (decaf::io::DataOutputStream &dataOut, const decaf::util::List< util::PrimitiveValueNode > &list) throw (decaf::io::IOException)
Marshal a List of Primitives to the given OutputStream, can result in recursive calls to this method if the list contains lists of lists.
- static void **marshalPrimitive** (decaf::io::DataOutputStream &dataOut, const util::PrimitiveValueNode &value) throw (decaf::io::IOException)
Used to Marshal the Primitive types out on the Wire.
- static void **unmarshalPrimitiveMap** (decaf::io::DataInputStream &dataIn, util::PrimitiveMap &map) throw (decaf::io::IOException)
Unmarshals a Map of Primitives from the given InputStream, can result in recursive calls to this method if the map contains maps of maps.
- static void **unmarshalPrimitiveList** (decaf::io::DataInputStream &dataIn, decaf::util::StlList< util::PrimitiveValueNode > &list) throw (decaf::io::IOException)
Unmarshals a List of Primitives from the given InputStream, can result in recursive calls to this method if the list contains lists of lists.
- static util::PrimitiveValueNode **unmarshalPrimitive** (decaf::io::DataInputStream &dataIn) throw (decaf::io::IOException)
Unmarshals a Primitive Type from the stream, and returns it as a value Node.

6.522.1 Detailed Description

This class wraps the functionality needed to marshal a primitive map to the Openwire Format's expectation of what the map looks like on the wire.

6.522.2 Constructor & Destructor Documentation

6.522.2.1 `activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::PrimitiveTypesMarshaller () [inline]`

6.522.2.2 `virtual activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::~~PrimitiveTypesMarshaller () [inline, virtual]`

6.522.3 Member Function Documentation

6.522.3.1 `static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::marshal (const util::PrimitiveMap * map, std::vector< unsigned char > & dest) throw (decaf::lang::Exception) [static]`

Static Marshal of a primitive map object.

Parameters

map Map to Marshal.

dest Reference to a byte array to house the data.

Exceptions

Exception

6.522.3.2 `static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::marshal (const util::PrimitiveList * list, std::vector< unsigned char > & dest) throw (decaf::lang::Exception) [static]`

Static Marshal of a primitive map object.

Parameters

list The list object to Marshal

dest Reference to a byte array to house the data

Exceptions

Exception

6.522.3.3 `static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::marshalPrimitive (decaf::io::DataOutputStream & dataOut, const util::PrimitiveValueNode & value) throw (decaf::io::IOException)` [static, protected]

Used to Marshal the Primitive types out on the Wire.

Parameters

dataOut - the DataOutputStream to write to

value - the ValueNode to write.

Exceptions

IOException

6.522.3.4 `static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::marshalPrimitiveList (decaf::io::DataOutputStream & dataOut, const decaf::util::List< util::PrimitiveValueNode > & list) throw (decaf::io::IOException)` [static, protected]

Marshal a List of Primitives to the given OutputStream, can result in recursive calls to this method if the list contains lists of lists.

Parameters

dataOut - the DataOutputStream to write to

list - the ValueNode to write.

Exceptions

IOException

6.522.3.5 `static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::marshalPrimitiveMap (decaf::io::DataOutputStream & dataOut, const decaf::util::Map< std::string, util::PrimitiveValueNode > & map) throw (decaf::io::IOException)` [static, protected]

Marshal a Map of Primitives to the given OutputStream, can result in recursive calls to this method if the map contains maps of maps.

Parameters

dataOut - the DataOutputStream to write to

map - the ValueNode to write.

Exceptions

IOException

6.522.3.6 static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::unmarshal (util::PrimitiveList * *list*, const std::vector< unsigned char > & *src*) throw (decaf::lang::Exception) [static]

Static Map Unmarshaler, takes an array of bytes and returns a new instance of a PrimitiveMap object.

Caller owns the pointer.

Parameters

list The list object to Un-marshal
src Reference to a byte array to read data from.

Exceptions

Exception

6.522.3.7 static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::unmarshal (util::PrimitiveMap * *map*, const std::vector< unsigned char > & *src*) throw (decaf::lang::Exception) [static]

Static Map Unmarshaler, takes an array of bytes and returns a new instance of a PrimitiveMap object.

Caller owns the pointer.

Parameters

map Map to Unmarshal into
src Reference to a byte array to read data from.

Exceptions

Exception

6.522.3.8 static util::PrimitiveValueNode activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::unmarshalPrimitive (decaf::io::DataInputStream & *dataIn*) throw (decaf::io::IOException) [static, protected]

Unmarshals a Primitive Type from the stream, and returns it as a value Node.

Parameters

dataIn - DataInputStream to read from.

Returns

a PrimitiveValueNode containing the data.

Exceptions

IOException

6.522.3.9 `static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::unmarshalPrimitiveList (decaf::io::DataInputStream & dataIn, decaf::util::StlList< util::PrimitiveValueNode > & list) throw (decaf::io::IOException)` [static, protected]

Unmarshals a List of Primitives from the given InputStream, can result in recursive calls to this method if the list contains lists of lists.

Parameters

dataIn - DataInputStream to read from.

list - the ValueNode to write.

Exceptions

IOException

6.522.3.10 `static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::unmarshalPrimitiveMap (decaf::io::DataInputStream & dataIn, util::PrimitiveMap & map) throw (decaf::io::IOException)` [static, protected]

Unmarshals a Map of Primitives from the given InputStream, can result in recursive calls to this method if the map contains maps of maps.

Parameters

dataIn - DataInputStream to read from.

map - the map to fill with data.

Exceptions

IOException

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/PrimitiveTypesMarshaller.h`

6.523 activemq::util::PrimitiveValueNode::PrimitiveValue Union Reference

Define a union type comprised of the various types.

```
#include <src/main/activemq/util/PrimitiveValueNode.h>
```

Data Fields

- bool **boolValue**
- unsigned char **byteValue**
- char **charValue**

- short **shortValue**
- int **intValue**
- long long **longValue**
- double **doubleValue**
- float **floatValue**
- std::string * **stringValue**
- std::vector< unsigned char > * **byteArrayValue**
- decaf::util::List< PrimitiveValueNode > * **listValue**
- decaf::util::Map< std::string, PrimitiveValueNode > * **mapValue**

6.523.1 Detailed Description

Define a union type comprised of the various types.

6.523.2 Field Documentation

- 6.523.2.1** bool **activemq::util::PrimitiveValueNode::PrimitiveValue::boolValue**
- 6.523.2.2** std::vector<unsigned char>* **activemq::util::PrimitiveValueNode::PrimitiveValue::byteArrayValue**
- 6.523.2.3** unsigned char **activemq::util::PrimitiveValueNode::PrimitiveValue::byteValue**
- 6.523.2.4** char **activemq::util::PrimitiveValueNode::PrimitiveValue::charValue**
- 6.523.2.5** double **activemq::util::PrimitiveValueNode::PrimitiveValue::doubleValue**
- 6.523.2.6** float **activemq::util::PrimitiveValueNode::PrimitiveValue::floatValue**
- 6.523.2.7** int **activemq::util::PrimitiveValueNode::PrimitiveValue::intValue**
- 6.523.2.8** decaf::util::List<PrimitiveValueNode>* **activemq::util::PrimitiveValueNode::PrimitiveValue::listValue**
- 6.523.2.9** long long **activemq::util::PrimitiveValueNode::PrimitiveValue::longValue**
- 6.523.2.10** decaf::util::Map<std::string, PrimitiveValueNode>* **activemq::util::PrimitiveValueNode::PrimitiveValue::mapValue**
- 6.523.2.11** short **activemq::util::PrimitiveValueNode::PrimitiveValue::shortValue**
- 6.523.2.12** std::string* **activemq::util::PrimitiveValueNode::PrimitiveValue::stringValue**

The documentation for this union was generated from the following file:

- src/main/activemq/util/**PrimitiveValueNode.h**

6.524 activemq::util::PrimitiveValueConverter Class Reference

Class controls the conversion of data contained in a **PrimitiveValueNode** (p. 2398) from one type to another.

```
#include <src/main/activemq/util/PrimitiveValueConverter.h>
```

Public Member Functions

- **PrimitiveValueConverter** ()
- virtual **~PrimitiveValueConverter** ()
- template<typename TO >
TO **convert** (const **PrimitiveValueNode** &value) const throw (decaf::lang::exceptions::UnsupportedOperationException)

6.524.1 Detailed Description

Class controls the conversion of data contained in a **PrimitiveValueNode** (p. 2398) from one type to another. If the conversion is supported then calling the convert method will throw an UnsupportedOperationException to indicate that its not possible to perform the conversion.

This class is used to implement the rules of conversion on CMS Message properties, the following conversion table must be implemented. A value written as the row type can be read in the column type.

		boolean	byte	short	int	long	float	double	String	-----														
boolean		X X	byte		X X X X X	short		X X X X	int		X X X	long		X X	float		X X X	double		X X	String		X X X X X X X X	-----

Since

3.0

6.524.2 Constructor & Destructor Documentation

6.524.2.1 **activemq::util::PrimitiveValueConverter::PrimitiveValueConverter** ()
[inline]

6.524.2.2 **virtual**
activemq::util::PrimitiveValueConverter::~~PrimitiveValueConverter ()
[inline, virtual]

6.524.3 Member Function Documentation

6.524.3.1 **std::vector< unsigned char > activemq::util::PrimitiveValueConverter::convert** (const **PrimitiveValueNode** & *value*) const throw (decaf::lang::exceptions::UnsupportedOperationException) [inline]

The documentation for this class was generated from the following file:

- src/main/activemq/util/**PrimitiveValueConverter.h**

6.525 activemq::util::PrimitiveValueNode Class Reference

Class that wraps around a single value of one of the many types.

```
#include <src/main/activemq/util/PrimitiveValueNode.h>
```

Data Structures

- union **PrimitiveValue**

Define a union type comprised of the various types.

Public Types

- enum **PrimitiveType** {
 NULL_TYPE = 0, **BOOLEAN_TYPE** = 1, **BYTE_TYPE** = 2, **CHAR_TYPE** = 3,
 SHORT_TYPE = 4, **INTEGER_TYPE** = 5, **LONG_TYPE** = 6, **DOUBLE_TYPE** = 7,
 FLOAT_TYPE = 8, **STRING_TYPE** = 9, **BYTE_ARRAY_TYPE** = 10, **MAP_TYPE** = 11,
 LIST_TYPE = 12, **BIG_STRING_TYPE** = 13 }

Enumeration for the various primitive types.

Public Member Functions

- **PrimitiveValueNode** ()
Default Constructor, creates a value of the NULL_TYPE.
- **PrimitiveValueNode** (bool value)
Boolean Value Constructor.
- **PrimitiveValueNode** (unsigned char value)
Byte Value Constructor.
- **PrimitiveValueNode** (char value)
Char Value Constructor.
- **PrimitiveValueNode** (short value)
Short Value Constructor.
- **PrimitiveValueNode** (int value)
Int Value Constructor.
- **PrimitiveValueNode** (long long value)
Long Value Constructor.
- **PrimitiveValueNode** (float value)

Float Value Constructor.

- **PrimitiveValueNode** (double value)
Double Value Constructor.
- **PrimitiveValueNode** (const char *value)
String Value Constructor.
- **PrimitiveValueNode** (const std::string &value)
String Value Constructor.
- **PrimitiveValueNode** (const std::vector< unsigned char > &value)
Byte Array Value Constructor.
- **PrimitiveValueNode** (const decaf::util::List< PrimitiveValueNode > &value)
Primitive List Constructor.
- **PrimitiveValueNode** (const decaf::util::Map< std::string, PrimitiveValueNode > &value)
Primitive Map Value Constructor.
- **PrimitiveValueNode** (const PrimitiveValueNode &node)
Copy constructor.
- **~PrimitiveValueNode** ()
- **PrimitiveValueNode & operator=** (const PrimitiveValueNode &node)
Assignment operator, copies the data from the other node.
- **bool operator==** (const PrimitiveValueNode &node) const
Comparison Operator, compares this node to the other node.
- **PrimitiveType getType** () const
Gets the Value Type of this type wrapper.
- **PrimitiveValue getValue** () const
Gets the internal Primitive Value object from this wrapper.
- **void setValue** (const PrimitiveValue &value, PrimitiveType valueType)
Sets the internal PrimitiveVale object to the new value along with the tag for the type that it consists of.
- **void clear** ()
Clears the value from this wrapper converting it back to a blank NULL_ TYPE value.
- **void setBool** (bool value)
Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.
- **bool getBool** () const throw (decaf::lang::exceptions::NoSuchElementException)
Gets the Boolean value of this Node.

- void **setByte** (unsigned char value)
Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.
- unsigned char **getByte** () const throw (decaf::lang::exceptions::NoSuchElementException)
Gets the Byte value of this Node.
- void **setChar** (char value)
Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.
- char **getChar** () const throw (decaf::lang::exceptions::NoSuchElementException)
Gets the Character value of this Node.
- void **setShort** (short value)
Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.
- short **getShort** () const throw (decaf::lang::exceptions::NoSuchElementException)
Gets the Short value of this Node.
- void **setInt** (int value)
Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.
- int **getInt** () const throw (decaf::lang::exceptions::NoSuchElementException)
Gets the Integer value of this Node.
- void **setLong** (long long value)
Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.
- long long **getLong** () const throw (decaf::lang::exceptions::NoSuchElementException)
Gets the Long value of this Node.
- void **setFloat** (float value)
Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.
- float **getFloat** () const throw (decaf::lang::exceptions::NoSuchElementException)
Gets the Float value of this Node.
- void **setDouble** (double value)
Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.
- double **getDouble** () const throw (decaf::lang::exceptions::NoSuchElementException)
Gets the Double value of this Node.

- void **setString** (const std::string &value)
Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.
- std::string **getString** () const throw (decaf::lang::exceptions::NoSuchElementException)
Gets the String value of this Node.
- void **setByteArray** (const std::vector< unsigned char > &value)
Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.
- std::vector< unsigned char > **getByteArray** () const throw (decaf::lang::exceptions::NoSuchElementException)
Gets the Byte Array value of this Node.
- void **setList** (const decaf::util::List< PrimitiveValueNode > &value)
Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.
- const decaf::util::List< PrimitiveValueNode > & **getList** () const throw (decaf::lang::exceptions::NoSuchElementException)
Gets the Primitive List value of this Node.
- void **setMap** (const decaf::util::Map< std::string, PrimitiveValueNode > &value)
Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.
- const decaf::util::Map< std::string, PrimitiveValueNode > & **getMap** () const throw (decaf::lang::exceptions::NoSuchElementException)
Gets the Primitive Map value of this Node.
- std::string **toString** () const
Creates a string representation of this value.

6.525.1 Detailed Description

Class that wraps around a single value of one of the many types. Manages memory for complex types, such as strings. Note: the destructor was left non-virtual so no virtual table will be created. This probably isn't necessary, but will avoid needless memory allocation. Since we'll never extend this class, not having a virtual destructor isn't a concern.

6.525.2 Member Enumeration Documentation

6.525.2.1 enum activemq::util::PrimitiveValueNode::PrimitiveType

Enumeration for the various primitive types.

Enumerator:

NULL_ TYPE

BOOLEAN_TYPE
BYTE_TYPE
CHAR_TYPE
SHORT_TYPE
INTEGER_TYPE
LONG_TYPE
DOUBLE_TYPE
FLOAT_TYPE
STRING_TYPE
BYTE_ARRAY_TYPE
MAP_TYPE
LIST_TYPE
BIG_STRING_TYPE

6.525.3 Constructor & Destructor Documentation

6.525.3.1 activemq::util::PrimitiveValueNode::PrimitiveValueNode ()

Default Constructor, creates a value of the `NULL_TYPE`.

6.525.3.2 activemq::util::PrimitiveValueNode::PrimitiveValueNode (bool *value*)

Boolean Value Constructor.

Parameters

value - the new value to store.

6.525.3.3 activemq::util::PrimitiveValueNode::PrimitiveValueNode (unsigned char *value*)

Byte Value Constructor.

Parameters

value - the new value to store.

6.525.3.4 activemq::util::PrimitiveValueNode::PrimitiveValueNode (char *value*)

Char Value Constructor.

Parameters

value - the new value to store.

6.525.3.5 activemq::util::PrimitiveValueNode::PrimitiveValueNode (short *value*)

Short Value Constructor.

Parameters

value - the new value to store.

6.525.3.6 activemq::util::PrimitiveValueNode::PrimitiveValueNode (int *value*)

Int Value Constructor.

Parameters

value - the new value to store.

6.525.3.7 activemq::util::PrimitiveValueNode::PrimitiveValueNode (long long *value*)

Long Value Constructor.

Parameters

value - the new value to store.

6.525.3.8 activemq::util::PrimitiveValueNode::PrimitiveValueNode (float *value*)

Float Value Constructor.

Parameters

value - the new value to store.

6.525.3.9 activemq::util::PrimitiveValueNode::PrimitiveValueNode (double *value*)

Double Value Constructor.

Parameters

value - the new value to store.

6.525.3.10 activemq::util::PrimitiveValueNode::PrimitiveValueNode (const char * *value*)

String Value Constructor.

Parameters

value - the new value to store.

6.525.3.11 `activemq::util::PrimitiveValueNode::PrimitiveValueNode (const std::string & value)`

String Value Constructor.

Parameters

value - the new value to store.

6.525.3.12 `activemq::util::PrimitiveValueNode::PrimitiveValueNode (const std::vector< unsigned char > & value)`

Byte Array Value Constructor.

Parameters

value - the new value to store.

6.525.3.13 `activemq::util::PrimitiveValueNode::PrimitiveValueNode (const decaf::util::List< PrimitiveValueNode > & value)`

Primitive List Constructor.

Parameters

value - the new value to store.

6.525.3.14 `activemq::util::PrimitiveValueNode::PrimitiveValueNode (const decaf::util::Map< std::string, PrimitiveValueNode > & value)`

Primitive Map Value Constructor.

Parameters

value - the new value to store.

6.525.3.15 `activemq::util::PrimitiveValueNode::PrimitiveValueNode (const PrimitiveValueNode & node)`

Copy constructor.

Parameters

node The instance of another node to copy to this one.

6.525.3.16 `activemq::util::PrimitiveValueNode::~~PrimitiveValueNode ()`
`[inline]`

6.525.4 Member Function Documentation

6.525.4.1 `void activemq::util::PrimitiveValueNode::clear ()`

Clears the value from this wrapper converting it back to a blank NULL_TYPE value.

6.525.4.2 `bool activemq::util::PrimitiveValueNode::getBool () const throw (decaf::lang::exceptions::NoSuchElementException)`

Gets the Boolean value of this Node.

Returns

value contained at the given index

Exceptions

NoSuchElementException this node cannot be returned as the requested type.

6.525.4.3 `unsigned char activemq::util::PrimitiveValueNode::getBytes () const throw (decaf::lang::exceptions::NoSuchElementException)`

Gets the Byte value of this Node.

Returns

value contained at the given index

Exceptions

NoSuchElementException this node cannot be returned as the requested type.

6.525.4.4 `std::vector<unsigned char> activemq::util::PrimitiveValueNode::getBytesArray () const throw (decaf::lang::exceptions::NoSuchElementException)`

Gets the Byte Array value of this Node.

Returns

value contained at the given index

Exceptions

NoSuchElementException this node cannot be returned as the requested type.

6.525.4.5 `char activemq::util::PrimitiveValueNode::getChar () const throw (decaf::lang::exceptions::NoSuchElementException)`

Gets the Character value of this Node.

Returns

value contained at the given index

Exceptions

NoSuchElementException this node cannot be returned as the requested type.

6.525.4.6 `double activemq::util::PrimitiveValueNode::getDouble () const throw (decaf::lang::exceptions::NoSuchElementException)`

Gets the Double value of this Node.

Returns

value contained at the given index

Exceptions

NoSuchElementException this node cannot be returned as the requested type.

6.525.4.7 `float activemq::util::PrimitiveValueNode::getFloat () const throw (decaf::lang::exceptions::NoSuchElementException)`

Gets the Float value of this Node.

Returns

value contained at the given index

Exceptions

NoSuchElementException this node cannot be returned as the requested type.

6.525.4.8 `int activemq::util::PrimitiveValueNode::getInt () const throw (decaf::lang::exceptions::NoSuchElementException)`

Gets the Integer value of this Node.

Returns

value contained at the given index

Exceptions

NoSuchElementException this node cannot be returned as the requested type.

6.525.4.9 `const decaf::util::List<PrimitiveValueNode>&
activemq::util::PrimitiveValueNode::getList () const throw (
decaf::lang::exceptions::NoSuchElementException)`

Gets the Primitive List value of this Node.

Returns

value contained at the given index

Exceptions

NoSuchElementException this node cannot be returned as the requested type.

6.525.4.10 `long long activemq::util::PrimitiveValueNode::getLong () const throw (
(decaf::lang::exceptions::NoSuchElementException)`

Gets the Long value of this Node.

Returns

value contained at the given index

Exceptions

NoSuchElementException this node cannot be returned as the requested type.

6.525.4.11 `const decaf::util::Map<std::string, PrimitiveValueNode>&
activemq::util::PrimitiveValueNode::getMap () const throw (
decaf::lang::exceptions::NoSuchElementException)`

Gets the Primitive Map value of this Node.

Returns

value contained at the given index

Exceptions

NoSuchElementException this node cannot be returned as the requested type.

6.525.4.12 `short activemq::util::PrimitiveValueNode::getShort () const throw (
decaf::lang::exceptions::NoSuchElementException)`

Gets the Short value of this Node.

Returns

value contained at the given index

Exceptions

NoSuchElementException this node cannot be returned as the requested type.

6.525.4.13 `std::string activemq::util::PrimitiveValueNode::getString () const`
`throw (decaf::lang::exceptions::NoSuchElementException)`

Gets the String value of this Node.

Returns

value contained at the given index

Exceptions

NoSuchElementException this node cannot be returned as the requested type.

6.525.4.14 `PrimitiveType activemq::util::PrimitiveValueNode::getType () const`
`[inline]`

Gets the Value Type of this type wrapper.

Returns

the PrimitiveType value for this wrapper.

6.525.4.15 `PrimitiveValue activemq::util::PrimitiveValueNode::getValue () const`
`[inline]`

Gets the internal Primitive Value object from this wrapper.

Returns

a copy of the contained **PrimitiveValue** (p. 2395)

6.525.4.16 `PrimitiveValueNode& activemq::util::PrimitiveValueNode::operator= (`
`const PrimitiveValueNode & node)`

Assignment operator, copies the data from the other node.

Parameters

node The instance of another node to copy to this one.

6.525.4.17 `bool activemq::util::PrimitiveValueNode::operator== (const`
`PrimitiveValueNode & node) const`

Comparison Operator, compares this node to the other node.

Returns

true if the values are the same false otherwise.

6.525.4.18 void activemq::util::PrimitiveValueNode::setBool (bool *value*)

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

Parameters

value - the new value to assign to the element at index

6.525.4.19 void activemq::util::PrimitiveValueNode::setByte (unsigned char *value*)

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

Parameters

value - the new value to assign to the element at index

6.525.4.20 void activemq::util::PrimitiveValueNode::setByteArray (const std::vector< unsigned char > & *value*)

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

Parameters

value - the new value to assign to the element at index

6.525.4.21 void activemq::util::PrimitiveValueNode::setChar (char *value*)

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

Parameters

value - the new value to assign to the element at index

6.525.4.22 void activemq::util::PrimitiveValueNode::setDouble (double *value*)

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

Parameters

value - the new value to assign to the element at index

6.525.4.23 void activemq::util::PrimitiveValueNode::setFloat (float *value*)

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

Parameters

value - the new value to assign to the element at index

6.525.4.24 void activemq::util::PrimitiveValueNode::setInt (int *value*)

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

Parameters

value - the new value to assign to the element at index

6.525.4.25 void activemq::util::PrimitiveValueNode::setList (const decaf::util::List< PrimitiveValueNode > & *value*)

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

Parameters

value - the new value to assign to the element at index

6.525.4.26 void activemq::util::PrimitiveValueNode::setLong (long long *value*)

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

Parameters

value - the new value to assign to the element at index

6.525.4.27 void activemq::util::PrimitiveValueNode::setMap (const decaf::util::Map< std::string, PrimitiveValueNode > & *value*)

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

Parameters

value - the new value to assign to the element at index

6.525.4.28 void activemq::util::PrimitiveValueNode::setShort (short *value*)

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

Parameters

value - the new value to assign to the element at index

6.525.4.29 void activemq::util::PrimitiveValueNode::setString (const std::string & *value*)

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

Parameters

value - the new value to assign to the element at index

6.525.4.30 void activemq::util::PrimitiveValueNode::setValue (const PrimitiveValue & *value*, PrimitiveType *valueType*)

Sets the internal PrimitiveVale object to the new value along with the tag for the type that it consists of.

Parameters

value The value to set as the value contained in this Node.

valueType The type of the value being set into this one.

6.525.4.31 std::string activemq::util::PrimitiveValueNode::toString () const

Creates a string representation of this value.

Returns

string value of this type wrapper.

The documentation for this class was generated from the following file:

- src/main/activemq/util/**PrimitiveValueNode.h**

6.526 decaf::security::Principal Class Reference

Base interface for a principal, which can represent an individual or organization.

#include <src/main/decaf/security/Principal.h>

Inheritance diagram for decaf::security::Principal:

Public Member Functions

- virtual `~Principal` ()
- virtual bool `equals` (const `Principal` &another) const =0
Compares two principals to see if they are the same.
- virtual std::string `getName` () const =0
Provides the name of this principal.

6.526.1 Detailed Description

Base interface for a principal, which can represent an individual or organization.

6.526.2 Constructor & Destructor Documentation

6.526.2.1 virtual decaf::security::Principal::~~Principal () [inline, virtual]

6.526.3 Member Function Documentation

6.526.3.1 virtual bool decaf::security::Principal::equals (const `Principal` &
another) const [pure virtual]

Compares two principals to see if they are the same.

Parameters

another A principal to be tested for equality to this one.

Returns

true if the given principal is equivalent to this one.

6.526.3.2 virtual std::string decaf::security::Principal::getName () const [pure virtual]

Provides the name of this principal.

Returns

the name of this principal.

Implemented in `decaf::security::auth::x500::X500Principal` (p. 3189).

The documentation for this class was generated from the following file:

- src/main/decaf/security/**Principal.h**

6.527 decaf::util::PriorityQueue< E > Class Template Reference

An unbounded priority queue based on a binary heap algorithm.

```
#include <src/main/decaf/util/PriorityQueue.h>
```

Inheritance diagram for decaf::util::PriorityQueue< E >:

Data Structures

- class **ConstPriorityQueueIterator**
- class **PriorityQueueIterator**

Public Member Functions

- **PriorityQueue** ()
*Creates a **Priority Queue** (p. 2507) with the default initial capacity.*
- **PriorityQueue** (std::size_t initialCapacity)
*Creates a **Priority Queue** (p. 2507) with the capacity value supplied.*
- **PriorityQueue** (std::size_t initialCapacity, **Comparator**< E > *comparator)
*Creates a **Priority Queue** (p. 2507) with the default initial capacity.*
- **PriorityQueue** (const **Collection**< E > &source)
*Creates a **PriorityQueue** (p. 2413) containing the elements in the specified **Collection** (p. 982).*
- **PriorityQueue** (const **PriorityQueue**< E > &source)
*Creates a **PriorityQueue** (p. 2413) containing the elements in the specified priority queue.*
- virtual ~**PriorityQueue** ()
- **PriorityQueue**< E > &operator= (const **Collection**< E > &source)
*Assignment operator, assign another **Collection** (p. 982) to this one.*
- **PriorityQueue**< E > &operator= (const **PriorityQueue**< E > &source)
*Assignment operator, assign another **PriorityQueue** (p. 2413) to this one.*
- virtual **decaf::util::Iterator**< E > *iterator ()
- virtual **decaf::util::Iterator**< E > *iterator () const
- virtual std::size_t size () const
Returns the number of elements in this collection.
- virtual void **clear** () throw (lang::exceptions::UnsupportedOperationException)
Removes all elements of the queue.
- virtual bool **offer** (const E &value) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException)

Inserts the specified element into the queue provided that the condition allows such an operation.

- virtual bool **poll** (E &result)
Gets and removes the element in the head of the queue.
- virtual bool **peek** (E &result) const
Gets but not removes the element in the head of the queue.
- virtual E **remove** () throw (decaf::lang::exceptions::NoSuchElementException)
Retrieves and removes the head of this queue.
- virtual bool **remove** (const E &value) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException)
Removes a single instance of the specified element from this collection, if it is present (optional operation).
- virtual bool **add** (const E &value) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException)
Inserts the specified element into this queue if it is possible to do so immediately without violating capacity restrictions, returning true upon success and throwing an IllegalStateException if no space is currently available.
- decaf::lang::Pointer< Comparator< E > > **comparator** () const
*obtains a Copy of the Pointer instance that this **PriorityQueue** (p. 2413) is using to compare the elements in the queue with.*

Friends

- class **PriorityQueueIterator**

6.527.1 Detailed Description

template<typename E> class decaf::util::PriorityQueue< E >

An unbounded priority queue based on a binary heap algorithm. The elements of the priority queue are ordered according to their natural ordering, or by a **Comparator** (p. 1011) provided to one of the constructors that accepts Comparators. A priority queue relying on natural ordering also does not permit insertion of non-comparable objects (doing so may result in a compiler error).

The head of this queue is the least element with respect to the specified ordering. If multiple elements are tied for least value, the head is one of those elements -- ties are broken arbitrarily. The queue retrieval operations poll, remove, peek, and element access the element at the head of the queue.

A priority queue is unbounded, but has an internal capacity governing the size of an array used to store the elements on the queue. It is always at least as large as the queue size. As elements are added to a priority queue, its capacity grows automatically. The details of the growth policy are not specified.

This class and its iterator implement all of the optional methods of the **Collection** (p. 982) and **Iterator** (p. 1716) interfaces. The **Iterator** (p. 1716) provided in method **iterator()** (p. 2417) is

not guaranteed to traverse the elements of the priority queue in any particular order. If you need ordered traversal, consider using `Arrays::sort(pq.toArray())`.

Note that this implementation is not synchronized. Multiple threads should not access a **PriorityQueue** (p. 2413) instance concurrently if any of the threads modifies the queue. Instead, use the thread-safe `PriorityBlockingQueue` class.

Implementation note: this implementation provides $O(\log(n))$ time for the enqueueing and dequeuing methods (`offer`, `poll`, **`remove()`** (p. 2419) and `add`); linear time for the `remove(Object)` and `contains(Object)` methods; and constant time for the retrieval methods (`peek`, `element`, and `size`).

Since

1.0

6.527.2 Constructor & Destructor Documentation

6.527.2.1 `template<typename E> decaf::util::PriorityQueue< E >::PriorityQueue () [inline]`

Creates a **Priority Queue** (p. 2507) with the default initial capacity.

6.527.2.2 `template<typename E> decaf::util::PriorityQueue< E >::PriorityQueue (std::size_t initialCapacity) [inline]`

Creates a **Priority Queue** (p. 2507) with the capacity value supplied.

Parameters

initialCapacity The initial number of elements allocated to this **PriorityQueue** (p. 2413).

6.527.2.3 `template<typename E> decaf::util::PriorityQueue< E >::PriorityQueue (std::size_t initialCapacity, Comparator< E > * comparator) [inline]`

Creates a **Priority Queue** (p. 2507) with the default initial capacity.

This new **PriorityQueue** (p. 2413) takes ownership of the passed **Comparator** (p. 1011) instance and uses that to determine the ordering of the elements in the **Queue** (p. 2507).

Parameters

initialCapacity The initial number of elements allocated to this **PriorityQueue** (p. 2413).

comparator The **Comparator** (p. 1011) instance to use in sorting the elements in the **Queue** (p. 2507).

Exceptions

NullPointerException if the passed **Comparator** (p. 1011) is NULL.

6.527.2.4 `template<typename E> decaf::util::PriorityQueue< E >::PriorityQueue (const Collection< E > & source) [inline]`

Creates a **PriorityQueue** (p. 2413) containing the elements in the specified **Collection** (p. 982).

Parameters

source the **Collection** (p. 982) whose elements are to be placed into this priority queue

6.527.2.5 `template<typename E> decaf::util::PriorityQueue< E >::PriorityQueue (const PriorityQueue< E > & source) [inline]`

Creates a **PriorityQueue** (p. 2413) containing the elements in the specified priority queue.

This priority queue will be ordered according to the same ordering as the given priority queue.

Parameters

source the priority queue whose elements are to be placed into this priority queue

6.527.2.6 `template<typename E> virtual decaf::util::PriorityQueue< E >::~~PriorityQueue () [inline, virtual]`

6.527.3 Member Function Documentation

6.527.3.1 `template<typename E> virtual bool decaf::util::PriorityQueue< E >::add (const E & value) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException) [inline, virtual]`

Inserts the specified element into this queue if it is possible to do so immediately without violating capacity restrictions, returning true upon success and throwing an `IllegalStateException` if no space is currently available.

This implementation returns true if offer succeeds, else throws an `IllegalStateException`.

Parameters

value - the element to offer to the **Queue** (p. 2507).

Returns

true if the add succeeds.

Exceptions

IllegalArgumentException if the element cannot be added.

Reimplemented from `decaf::util::AbstractQueue< E >` (p. 135).

References `DECAF_CATCH_EXCEPTION_CONVERT`, `DECAF_CATCH_RETHROW`, `DECAF_CATCHALL_THROW`, and `decaf::util::PriorityQueue< E >::offer()`.

6.527.3.2 `template<typename E> virtual void decaf::util::PriorityQueue< E >::clear () throw (lang::exceptions::UnsupportedOperationException) [inline, virtual]`

Removes all elements of the queue.

This implementation repeatedly invokes poll until it returns the empty marker.

Reimplemented from `decaf::util::AbstractQueue< E >` (p. 136).

6.527.3.3 `template<typename E> decaf::lang::Pointer< Comparator<E> > decaf::util::PriorityQueue< E >::comparator () const [inline]`

obtains a Copy of the Pointer instance that this **PriorityQueue** (p. 2413) is using to compare the elements in the queue with.

The returned value is a copy, the caller cannot change the value if the internal Pointer value.

Returns

a copy of the **Comparator** (p. 1011) Pointer being used by this **Queue** (p. 2507).

6.527.3.4 `template<typename E> virtual decaf::util::Iterator<E>* decaf::util::PriorityQueue< E >::iterator () const [inline, virtual]`

6.527.3.5 `template<typename E> virtual decaf::util::Iterator<E>* decaf::util::PriorityQueue< E >::iterator () [inline, virtual]`

References `decaf::util::PriorityQueue< E >::PriorityQueueIterator`.

6.527.3.6 `template<typename E> virtual bool decaf::util::PriorityQueue< E >::offer (const E & value) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException) [inline, virtual]`

Inserts the specified element into the queue provided that the condition allows such an operation.

The method is generally preferable to the `collection.add(E)`, since the latter might throw an exception if the operation fails.

Parameters

value the specified element to insert into the queue.

Returns

true if the operation succeeds and false if it fails.

Exceptions

NullPointerException if the **Queue** (p. 2507) implementation does not allow Null values to be inserted into the **Queue** (p. 2507).

IllegalArgumentException if some property of the specified element prevents it from being added to this queue

Implements **decaf::util::Queue< E >** (p. 2509).

Referenced by **decaf::util::PriorityQueue< E >::add()**.

6.527.3.7 `template<typename E> PriorityQueue<E>& decaf::util::PriorityQueue<E>::operator= (const PriorityQueue< E > & source) [inline]`

Assignment operator, assign another **PriorityQueue** (p. 2413) to this one.

Parameters

source The **PriorityQueue** (p. 2413) to copy to this one.

6.527.3.8 `template<typename E> PriorityQueue<E>& decaf::util::PriorityQueue<E>::operator= (const Collection< E > & source) [inline]`

Assignment operator, assign another **Collection** (p. 982) to this one.

Parameters

source The **Collection** (p. 982) to copy to this one.

6.527.3.9 `template<typename E> virtual bool decaf::util::PriorityQueue< E>::peek (E & result) const [inline, virtual]`

Gets but not removes the element in the head of the queue.

The result if successful is assigned to the result parameter.

Parameters

result Reference to an instance of the contained type to assigned the removed value to.

Returns

true if the element at the head of the queue was removed and assigned to the result parameter.

Implements **decaf::util::Queue< E >** (p. 2509).

6.527.3.10 `template<typename E> virtual bool decaf::util::PriorityQueue< E>::poll (E & result) [inline, virtual]`

Gets and removes the element in the head of the queue.

If the operation succeeds the value of the element at the head of the **Queue** (p. 2507) is assigned to the result parameter and the method returns true. If the operation fails the method returns false and the value of the result parameter is undefined.

Parameters

result Reference to an instance of the contained type to assigned the removed value to.

Returns

true if the element at the head of the queue was removed and assigned to the result parameter.

Implements **decaf::util::Queue< E >** (p. 2509).

6.527.3.11 `template<typename E> virtual E decaf::util::PriorityQueue< E >::remove () throw (decaf::lang::exceptions::NoSuchElementException) [inline, virtual]`

Retrieves and removes the head of this queue.

This method differs from poll only in that it throws an exception if this queue is empty.

This implementation returns the result of poll unless the queue is empty.

Returns

a copy of the element in the head of the queue.

Exceptions

NoSuchElementException if the queue is empty.

Reimplemented from `decaf::util::AbstractQueue< E >` (p. 136).

6.527.3.12 `template<typename E> virtual bool decaf::util::PriorityQueue< E >::remove (const E & value) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException) [inline, virtual]`

Removes a single instance of the specified element from this collection, if it is present (optional operation).

More formally, removes the first element *e* such that *e* == *o*, if this collection contains one or more such elements. Returns true if this collection contained the specified element (or equivalently, if this collection changed as a result of the call).

This implementation iterates over the collection looking for the specified element. If it finds the element, it removes the element from the collection using the iterator's remove method.

Note that this implementation throws an `UnsupportedOperationException` if the iterator returned by this collection's iterator method does not implement the remove method and this collection contains the specified object.

Parameters

value - element to be removed from this collection, if present

Returns

true if an element was removed as a result of this call

Exceptions

UnsupportedOperationException if the remove operation is not supported by this collection.

IllegalArgumentException If the value is not a valid entry for this **Collection** (p. 982).

Reimplemented from `decaf::util::AbstractCollection< E >` (p. 127).

6.527.3.13 `template<typename E> virtual std::size_t decaf::util::PriorityQueue<E>::size () const [inline, virtual]`

Returns the number of elements in this collection.

If this collection contains more than Integer.MAX_VALUE elements, returns Integer.MAX_VALUE.

Returns

the number of elements in this collection

Implements `decaf::util::Collection< E >` (p. 990).

6.527.4 Friends And Related Function Documentation

6.527.4.1 `template<typename E> friend class PriorityQueueIterator [friend]`

Referenced by `decaf::util::PriorityQueue< E >::iterator()`.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/ProducerAck.h`

6.528 activemq::commands::ProducerAck Class Reference

```
#include <src/main/activemq/commands/ProducerAck.h>
```

Inheritance diagram for `activemq::commands::ProducerAck`:

Public Member Functions

- **ProducerAck** ()
- virtual **~ProducerAck** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **ProducerAck * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1372) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent.*

- virtual const **Pointer**< **ProducerId** > & **getProducerId** () const
- virtual **Pointer**< **ProducerId** > & **getProducerId** ()
- virtual void **setProducerId** (const **Pointer**< **ProducerId** > &producerId)
- virtual int **getSize** () const
- virtual void **setSize** (int size)
- virtual bool **isProducerAck** () const
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor *visitor)
throw (exceptions::ActiveMQException)

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_PRODUCERACK** = 19

Protected Member Functions

- **ProducerAck** (const **ProducerAck** &)
- **ProducerAck** & **operator=** (const **ProducerAck** &)

Protected Attributes

- **Pointer**< **ProducerId** > **producerId**
- int **size**

6.528.1 Constructor & Destructor Documentation

6.528.1.1 **activemq::commands::ProducerAck::ProducerAck** (const **ProducerAck** &) [inline, protected]

6.528.1.2 **activemq::commands::ProducerAck::ProducerAck** ()

6.528.1.3 **virtual activemq::commands::ProducerAck::~~ProducerAck** ()
[virtual]

6.528.2 Member Function Documentation

6.528.2.1 **virtual ProducerAck* activemq::commands::ProducerAck::cloneDataStructure** () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1373).

6.528.2.2 `virtual void activemq::commands::ProducerAck::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

Reimplemented from `activemq::commands::BaseCommand` (p. 598).

6.528.2.3 `virtual bool activemq::commands::ProducerAck::equals (const DataStructure * value) const [virtual]`

Compares the `DataStructure` (p. 1372) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 598).

6.528.2.4 `virtual unsigned char activemq::commands::ProducerAck::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

Returns

new `DataStructure` (p. 1372) type copy.

Implements `activemq::commands::DataStructure` (p. 1375).

6.528.2.5 `virtual Pointer<ProducerId>& activemq::commands::ProducerAck::getProducerId () [virtual]`

6.528.2.6 `virtual const Pointer<ProducerId>& activemq::commands::ProducerAck::getProducerId () const [virtual]`

6.528.2.7 `virtual int activemq::commands::ProducerAck::getSize () const [virtual]`

6.528.2.8 `virtual bool activemq::commands::ProducerAck::isProducerAck () const [inline, virtual]`

Returns

an answer of true to the `isProducerAck()` (p. 2422) query.

Reimplemented from `activemq::commands::BaseCommand` (p. 601).

- 6.528.2.9** `ProducerAck& activemq::commands::ProducerAck::operator= (const ProducerAck &)` [inline, protected]
- 6.528.2.10** `virtual void activemq::commands::ProducerAck::setProducerId (const Pointer< ProducerId > & producerId)` [virtual]
- 6.528.2.11** `virtual void activemq::commands::ProducerAck::setSize (int size)` [virtual]
- 6.528.2.12** `virtual std::string activemq::commands::ProducerAck::toString () const` [virtual]

Returns a string containing the information for this **DataStructure** (p.1372) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 602).

- 6.528.2.13** `virtual Pointer<Command> activemq::commands::ProducerAck::visit (activemq::state::CommandVisitor * visitor) throw (exceptions::ActiveMQException)` [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

Returns

a **Response** (p. 2611) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p.995).

6.528.3 Field Documentation

- 6.528.3.1** `const unsigned char activemq::commands::ProducerAck::ID_-PRODUCERACK = 19` [static]
- 6.528.3.2** `Pointer<ProducerId> activemq::commands::ProducerAck::producerId` [protected]
- 6.528.3.3** `int activemq::commands::ProducerAck::size` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ProducerAck.h`

6.529 activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller Class Reference

Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 2424).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ProducerAckMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller:

Public Member Functions

- **ProducerAckMarshaller** ()
- virtual **~ProducerAckMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.529.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 2424). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.529.2 Constructor & Destructor Documentation

6.529.2.1 `activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller::ProducerAckMarshaller () [inline]`

6.529.2.2 `virtual activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller::~~ProducerAckMarshaller () [inline, virtual]`

6.529.3 Member Function Documentation

6.529.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.529.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.529.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 631).

6.529.3.4 virtual void activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 632).

6.529.3.5 virtual int activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 633).

6.529.3.6 virtual void activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 634).

```
6.529.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 635).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ProducerAckMarshaller.h`

6.530 `activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller` Class Reference

Marshaling code for Open Wire Format for `ProducerAckMarshaller` (p. 2427).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ProducerAckMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller`:

Public Member Functions

- **ProducerAckMarshaller** ()
- virtual **~ProducerAckMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.530.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p.2427). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.530.2 Constructor & Destructor Documentation

6.530.2.1 `activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller::ProducerAckMarshaller () [inline]`

6.530.2.2 `virtual activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller::~~ProducerAckMarshaller () [inline, virtual]`

6.530.3 Member Function Documentation

6.530.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.530.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.530.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 604).

6.530.3.4 virtual void activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 605).

6.530.3.5 virtual int activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 606).

6.530.3.6 virtual void activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 608).

```
6.530.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 609).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ProducerAckMarshaller.h`

6.531 `activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller` Class Reference

Marshaling code for Open Wire Format for `ProducerAckMarshaller` (p. 2431).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ProducerAckMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller`:

Public Member Functions

- **ProducerAckMarshaller** ()
- virtual **~ProducerAckMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.531.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p.2431). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.531.2 Constructor & Destructor Documentation

6.531.2.1 `activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller::ProducerAckMarshaller () [inline]`

6.531.2.2 `virtual activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller::~~ProducerAckMarshaller () [inline, virtual]`

6.531.3 Member Function Documentation

6.531.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.531.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.531.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 618).

6.531.3.4 virtual void activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 619).

6.531.3.5 virtual int activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 620).

6.531.3.6 virtual void activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 621).

```
6.531.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 622).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/ProducerAckMarshaller.h`

6.532 **activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller** Class Reference

Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 2435).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ProducerAckMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller**:

Public Member Functions

- **ProducerAckMarshaller** ()
- virtual **~ProducerAckMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.532.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p.2435). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.532.2 Constructor & Destructor Documentation

6.532.2.1 `activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller::ProducerAckMarshaller () [inline]`

6.532.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller::~~ProducerAckMarshaller () [inline, virtual]`

6.532.3 Member Function Documentation

6.532.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.532.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.532.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 624).

6.532.3.4 virtual void activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 625).

6.532.3.5 virtual int activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 626).

6.532.3.6 virtual void activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 628).

```
6.532.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 629).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ProducerAckMarshaller.h`

6.533 **activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller** Class Reference

Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 2439).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ProducerAckMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller**:

Public Member Functions

- **ProducerAckMarshaller** ()
- virtual **~ProducerAckMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.533.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p.2439). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.533.2 Constructor & Destructor Documentation

6.533.2.1 `activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller::ProducerAckMarshaller () [inline]`

6.533.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller::~~ProducerAckMarshaller () [inline, virtual]`

6.533.3 Member Function Documentation

6.533.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.533.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.533.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 611).

6.533.3.4 virtual void activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 612).

6.533.3.5 virtual int activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 613).

6.533.3.6 virtual void activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 614).

```
6.533.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 615).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/ProducerAckMarshaller.h`

6.534 activemq::cmsutil::ProducerCallback Class Reference

Callback for sending a message to a CMS destination.

```
#include <src/main/activemq/cmsutil/ProducerCallback.h>
```

Inheritance diagram for `activemq::cmsutil::ProducerCallback`:

Public Member Functions

- virtual `~ProducerCallback` ()
- virtual void `doInCms` (`cms::Session` *session, `cms::MessageProducer` *producer)=0
throw (`cms::CMSEException`)

Execute an action given a session and producer.

6.534.1 Detailed Description

Callback for sending a message to a CMS destination.

6.534.2 Constructor & Destructor Documentation

- 6.534.2.1** virtual `activemq::cmsutil::ProducerCallback::~~ProducerCallback` ()
[inline, virtual]

6.534.3 Member Function Documentation

- 6.534.3.1** virtual void `activemq::cmsutil::ProducerCallback::doInCms` (
`cms::Session` * *session*, `cms::MessageProducer` * *producer*) throw (
`cms::CMSEException`) [pure virtual]

Execute an action given a session and producer.

Parameters

session the CMS Session

producer the CMS Producer

Exceptions

cms::CMSEException (p. 960) if thrown by CMS API methods

Implemented in `activemq::cmsutil::CmsTemplate::SendExecutor` (p. 2661).

The documentation for this class was generated from the following file:

- `src/main/activemq/cmsutil/ProducerCallback.h`

6.535 activemq::cmsutil::CmsTemplate::ProducerExecutor Class Reference

```
#include <src/main/activemq/cmsutil/CmsTemplate.h>
```

Inheritance diagram for `activemq::cmsutil::CmsTemplate::ProducerExecutor`:

Public Member Functions

- **ProducerExecutor** (**ProducerCallback** **action*, **CmsTemplate** **parent*, **cms::Destination** **destination*)
- virtual **~ProducerExecutor** ()
- virtual void **doInCms** (**cms::Session** **session*) throw (**cms::CMSEException**)

Execute any number of operations against the supplied CMS session.

- virtual **cms::Destination** * **getDestination** (**cms::Session** **session* **AMQCPP_UNUSED**) throw (**cms::CMSEException**)

Protected Attributes

- **ProducerCallback** * *action*
- **CmsTemplate** * *parent*
- **cms::Destination** * *destination*

6.535.1 Constructor & Destructor Documentation

6.535.1.1 **activemq::cmsutil::CmsTemplate::ProducerExecutor::ProducerExecutor** (**ProducerCallback** * *action*, **CmsTemplate** * *parent*, **cms::Destination** * *destination*) [inline]

6.535.1.2 virtual **activemq::cmsutil::CmsTemplate::ProducerExecutor::~~ProducerExecutor** () [inline, virtual]

6.535.2 Member Function Documentation

6.535.2.1 virtual void **activemq::cmsutil::CmsTemplate::ProducerExecutor::doInCms** (**cms::Session** * *session*) throw (**cms::CMSEException**) [virtual]

Execute any number of operations against the supplied CMS session.

Parameters

session the CMS Session

Exceptions

cms::CMSEException (p. 960) if thrown by CMS API methods

Implements **activemq::cmsutil::SessionCallback** (p. 2677).

6.535.2.2 virtual cms::Destination* activemq::cmsutil::CmsTemplate::ProducerExecutor::getDestination (cms::Session *session *AMQCPP_UNUSED*) throw (cms::CMSException) [inline, virtual]

6.535.3 Field Documentation

6.535.3.1 ProducerCallback* activemq::cmsutil::CmsTemplate::ProducerExecutor::action [protected]

6.535.3.2 cms::Destination* activemq::cmsutil::CmsTemplate::ProducerExecutor::destination [protected]

6.535.3.3 CmsTemplate* activemq::cmsutil::CmsTemplate::ProducerExecutor::parent [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/CmsTemplate.h

6.536 activemq::commands::ProducerId Class Reference

```
#include <src/main/activemq/commands/ProducerId.h>
```

Inheritance diagram for activemq::commands::ProducerId:

Public Types

- typedef decaf::lang::PointerComparator< ProducerId > COMPARATOR

Public Member Functions

- **ProducerId** ()
- **ProducerId** (const **ProducerId** &other)
- **ProducerId** (const **SessionId** &sessionId, long long consumerId)
- virtual ~**ProducerId** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **ProducerId** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.

6.536.2.4 `virtual activemq::commands::ProducerId::~~ProducerId () [virtual]`

6.536.3 Member Function Documentation

6.536.3.1 `virtual ProducerId* activemq::commands::ProducerId::cloneDataStructure () const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

6.536.3.2 `virtual int activemq::commands::ProducerId::compareTo (const ProducerId & value) const [virtual]`

6.536.3.3 `virtual void activemq::commands::ProducerId::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

6.536.3.4 `virtual bool activemq::commands::ProducerId::equals (const DataStructure * value) const [virtual]`

Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

6.536.3.5 `virtual bool activemq::commands::ProducerId::equals (const ProducerId & value) const [virtual]`

6.536.3.6 `virtual std::string& activemq::commands::ProducerId::getConnectionId () [virtual]`

6.536.3.7 `virtual const std::string& activemq::commands::ProducerId::getConnectionId () const [virtual]`

6.536.3.8 `virtual unsigned char activemq::commands::ProducerId::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataSet** (p. 1372) type copy.

- 6.536.3.9** `const Pointer<SessionId>& activemq::commands::ProducerId::getParentId () const`
- 6.536.3.10** `virtual long long activemq::commands::ProducerId::getSessionId () const [virtual]`
- 6.536.3.11** `virtual long long activemq::commands::ProducerId::getValue () const [virtual]`
- 6.536.3.12** `virtual bool activemq::commands::ProducerId::operator< (const ProducerId & value) const [virtual]`
- 6.536.3.13** `ProducerId& activemq::commands::ProducerId::operator= (const ProducerId & other)`
- 6.536.3.14** `virtual bool activemq::commands::ProducerId::operator== (const ProducerId & value) const [virtual]`
- 6.536.3.15** `virtual void activemq::commands::ProducerId::setConnectionId (const std::string & connectionId) [virtual]`
- 6.536.3.16** `virtual void activemq::commands::ProducerId::setSessionId (long long sessionId) [virtual]`
- 6.536.3.17** `virtual void activemq::commands::ProducerId::setValue (long long value) [virtual]`
- 6.536.3.18** `virtual std::string activemq::commands::ProducerId::toString () const [virtual]`

Returns a string containing the information for this **DataSet** (p. 1372) such as its type and value of its elements.

Returns

formatted string useful for debugging.

6.536.4 Field Documentation

- 6.536.4.1** `std::string activemq::commands::ProducerId::connectionId [protected]`
- 6.536.4.2** `const unsigned char activemq::commands::ProducerId::ID__ - PRODUCERID = 123 [static]`

Referenced by `activemq::state::CommandVisitorAdapter::processRemoveInfo()`.

6.536.4.3 long long activemq::commands::ProducerId::sessionId [protected]

6.536.4.4 long long activemq::commands::ProducerId::value [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**ProducerId.h**

6.537 activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller Class Reference

Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 2450).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ProducerIdMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller:

Public Member Functions

- **ProducerIdMarshaller** ()
- virtual **~ProducerIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.537.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 2450). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.537.2 Constructor & Destructor Documentation

6.537.2.1 activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller::ProducerIdMarshaller () [inline]

6.537.2.2 virtual activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller::~~ProducerIdMarshaller () [inline, virtual]

6.537.3 Member Function Documentation

6.537.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1331).

6.537.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1336).

6.537.3.3 virtual void activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1342).

6.537.3.4 virtual void activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1348).

6.537.3.5 virtual int activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1354).

```
6.537.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1360).

```
6.537.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1366).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ProducerIdMarshaller.h`

6.538 **activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller** Class Reference

Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 2454).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ProducerIdMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller**:

Public Member Functions

- **ProducerIdMarshaller** ()
- virtual **~ProducerIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.538.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 2454). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.538.2 Constructor & Destructor Documentation

6.538.2.1 `activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller::ProducerIdMarshaller()` [inline]

6.538.2.2 `virtual activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller::~~ProducerIdMarshaller()` [inline, virtual]

6.538.3 Member Function Documentation

6.538.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.538.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.538.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1342).

```
6.538.3.4 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1348).

```
6.538.3.5 virtual int ac-
tivemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller::tightMarshall
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, utils::BooleanStream * bs ) throw (
decaf::io::IOException ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1354).

6.538.3.6 `virtual void activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1360).

6.538.3.7 `virtual void activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1366).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ProducerIdMarshaller.h`

6.539 `activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller` Class Reference

Marshaling code for Open Wire Format for `ProducerIdMarshaller` (p. 2457).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ProducerIdMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller:

Public Member Functions

- **ProducerIdMarshaller** ()
- virtual **~ProducerIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.539.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 2457). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.539.2 Constructor & Destructor Documentation

6.539.2.1 `activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller::ProducerIdMarshaller () [inline]`

6.539.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller::~~ProducerIdMarshaller () [inline, virtual]`

6.539.3 Member Function Documentation

6.539.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.539.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.539.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1342).

6.539.3.4 virtual void activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1348).

6.539.3.5 virtual int activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1354).

6.539.3.6 virtual void activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1360).

6.539.3.7 virtual void **activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller::tightUnmarshal**
 (**OpenWireFormat** * *wireFormat*, **commands::DataStructure**
 * *dataStructure*, **decaf::io::DataInputStream** * *dataIn*,
utils::BooleanStream * *bs*) throw (**decaf::io::IOException**) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1366).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ProducerIdMarshaller.h`

6.540 **activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller** Class Reference

Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 2461).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ProducerIdMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller**:

Public Member Functions

- **ProducerIdMarshaller** ()
- virtual **~ProducerIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.540.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 2461). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.540.2 Constructor & Destructor Documentation

6.540.2.1 `activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller::ProducerIdMarshaller () [inline]`

6.540.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller::~~ProducerIdMarshaller () [inline, virtual]`

6.540.3 Member Function Documentation

6.540.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.540.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.540.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1342).

6.540.3.4 virtual void activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1348).

6.540.3.5 virtual int activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1354).

6.540.3.6 virtual void activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1360).

```
6.540.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1366).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/ProducerIdMarshaller.h`

6.541 **activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller** Class Reference

Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 2465).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ProducerIdMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller**:

Public Member Functions

- **ProducerIdMarshaller** ()
- virtual **~ProducerIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.541.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 2465). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.541.2 Constructor & Destructor Documentation

6.541.2.1 `activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller::ProducerIdMarshaller () [inline]`

6.541.2.2 `virtual activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller::~~ProducerIdMarshaller () [inline, virtual]`

6.541.3 Member Function Documentation

6.541.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.541.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.541.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1342).

6.541.3.4 virtual void activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1348).

6.541.3.5 virtual int activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1354).

6.541.3.6 virtual void activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1360).

6.541.3.7 `virtual void activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1366).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/ProducerIdMarshaller.h`

6.542 activemq::commands::ProducerInfo Class Reference

```
#include <src/main/activemq/commands/ProducerInfo.h>
```

Inheritance diagram for `activemq::commands::ProducerInfo`:

Public Member Functions

- `ProducerInfo ()`
- `virtual ~ProducerInfo ()`
- `virtual unsigned char getDataStructureType () const`

Get the unique identifier that this object and its own Marshaler share.

- virtual **ProducerInfo** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1372) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **ProducerId** > & **getProducerId** () const
- virtual **Pointer**< **ProducerId** > & **getProducerId** ()
- virtual void **setProducerId** (const **Pointer**< **ProducerId** > &producerId)
- virtual const **Pointer**< **ActiveMQDestination** > & **getDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &destination)
- virtual const std::vector< **decaf::lang::Pointer**< **BrokerId** > > & **getBrokerPath** () const
- virtual std::vector< **decaf::lang::Pointer**< **BrokerId** > > & **getBrokerPath** ()
- virtual void **setBrokerPath** (const std::vector< **decaf::lang::Pointer**< **BrokerId** > > &brokerPath)
- virtual bool **isDispatchAsync** () const
- virtual void **setDispatchAsync** (bool dispatchAsync)
- virtual int **getWindowSize** () const
- virtual void **setWindowSize** (int windowSize)
- virtual bool **isProducerInfo** () const
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor *visitor) throw (exceptions::ActiveMQException)
Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_PRODUCERINFO** = 6

Protected Member Functions

- **ProducerInfo** (const **ProducerInfo** &)
- **ProducerInfo** & **operator=** (const **ProducerInfo** &)

Protected Attributes

- **Pointer< ProducerId > producerId**
- **Pointer< ActiveMQDestination > destination**
- **std::vector< decaf::lang::Pointer< BrokerId > > brokerPath**
- **bool dispatchAsync**
- **int windowSize**

6.542.1 Constructor & Destructor Documentation

6.542.1.1 **activemq::commands::ProducerInfo::ProducerInfo (const ProducerInfo &)** [inline, protected]

6.542.1.2 **activemq::commands::ProducerInfo::ProducerInfo ()**

6.542.1.3 **virtual activemq::commands::ProducerInfo::~~ProducerInfo ()** [virtual]

6.542.2 Member Function Documentation

6.542.2.1 **virtual ProducerInfo* activemq::commands::ProducerInfo::cloneDataStructure () const** [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1373).

6.542.2.2 **virtual void activemq::commands::ProducerInfo::copyDataStructure (const DataStructure * src)** [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 598).

6.542.2.3 **virtual bool activemq::commands::ProducerInfo::equals (const DataStructure * value) const** [virtual]

Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 598).

6.542.2.4 `virtual const std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::ProducerInfo::getBrokerPath () const [virtual]`

6.542.2.5 `virtual std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::ProducerInfo::getBrokerPath () [virtual]`

6.542.2.6 `virtual unsigned char activemq::commands::ProducerInfo::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataStructure** (p. 1372) type copy.

Implements **activemq::commands::DataStructure** (p. 1375).

6.542.2.7 `virtual const Pointer<ActiveMQDestination>& activemq::commands::ProducerInfo::getDestination () const [virtual]`

6.542.2.8 `virtual Pointer<ActiveMQDestination>& activemq::commands::ProducerInfo::getDestination () [virtual]`

6.542.2.9 `virtual Pointer<ProducerId>& activemq::commands::ProducerInfo::getProducerId () [virtual]`

6.542.2.10 `virtual const Pointer<ProducerId>& activemq::commands::ProducerInfo::getProducerId () const [virtual]`

6.542.2.11 `virtual int activemq::commands::ProducerInfo::getWindowSize () const [virtual]`

6.542.2.12 `virtual bool activemq::commands::ProducerInfo::isDispatchAsync () const [virtual]`

6.542.2.13 `virtual bool activemq::commands::ProducerInfo::isProducerInfo () const [inline, virtual]`

Returns

an answer of true to the **isProducerInfo()** (p. 2472) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 601).

- 6.542.2.14 `ProducerInfo& activemq::commands::ProducerInfo::operator= (const ProducerInfo &) [inline, protected]`
- 6.542.2.15 `virtual void activemq::commands::ProducerInfo::setBrokerPath (const std::vector< decaf::lang::Pointer< BrokerId > > & brokerPath) [virtual]`
- 6.542.2.16 `virtual void activemq::commands::ProducerInfo::setDestination (const Pointer< ActiveMQDestination > & destination) [virtual]`
- 6.542.2.17 `virtual void activemq::commands::ProducerInfo::setDispatchAsync (bool dispatchAsync) [virtual]`
- 6.542.2.18 `virtual void activemq::commands::ProducerInfo::setProducerId (const Pointer< ProducerId > & producerId) [virtual]`
- 6.542.2.19 `virtual void activemq::commands::ProducerInfo::setWindowSize (int windowSize) [virtual]`
- 6.542.2.20 `virtual std::string activemq::commands::ProducerInfo::toString () const [virtual]`

Returns a string containing the information for this **DataStructure** (p.1372) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 602).

- 6.542.2.21 `virtual Pointer<Command> activemq::commands::ProducerInfo::visit (activemq::state::CommandVisitor * visitor) throw (exceptions::ActiveMQException) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

Returns

a **Response** (p. 2611) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p.995).

6.542.3 Field Documentation

- 6.542.3.1 `std::vector< decaf::lang::Pointer<BrokerId> >`
`activemq::commands::ProducerInfo::brokerPath` [protected]
- 6.542.3.2 `Pointer<ActiveMQDestination>` `activemq::commands::ProducerInfo::destination`
[protected]
- 6.542.3.3 `bool` `activemq::commands::ProducerInfo::dispatchAsync` [protected]
- 6.542.3.4 `const unsigned char` `activemq::commands::ProducerInfo::ID_ - PRODUCERINFO = 6` [static]
- 6.542.3.5 `Pointer<ProducerId>` `activemq::commands::ProducerInfo::producerId`
[protected]
- 6.542.3.6 `int` `activemq::commands::ProducerInfo::windowSize` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ProducerInfo.h`

6.543 activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 2474).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ProducerInfoMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller`:

Public Member Functions

- **ProducerInfoMarshaller** ()
- virtual **~ProducerInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.543.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 2474). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.543.2 Constructor & Destructor Documentation

6.543.2.1 **activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller::ProducerInfoMarshaller** () [inline]

6.543.2.2 **virtual**
activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller::~~ProducerInfoMarshaller () [inline, virtual]

6.543.3 Member Function Documentation

6.543.3.1 **virtual commands::DataStructure*** **activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller::createObject** () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1331).

6.543.3.2 **virtual unsigned char** **activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller::getDataStructureType** () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.543.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 631).

6.543.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 632).

6.543.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 633).

```
6.543.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 634).

```
6.543.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 635).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ProducerInfoMarshaller.h`

6.544 **activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller** Class Reference

Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 2478).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ProducerInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller**:

Public Member Functions

- **ProducerInfoMarshaller** ()
- virtual **~ProducerInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)`

Write a object instance to data output stream.

6.544.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 2478). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.544.2 Constructor & Destructor Documentation

6.544.2.1 `activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller::ProducerInfoMarshaller () [inline]`

6.544.2.2 `virtual activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller::~~ProducerInfoMarshaller () [inline, virtual]`

6.544.3 Member Function Documentation

6.544.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.544.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.544.3.3 virtual void activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 604).

6.544.3.4 virtual void activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 605).

6.544.3.5 virtual int activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 606).

```
6.544.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 608).

```
6.544.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 609).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**ProducerInfoMarshaller.h**

6.545 activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 2482).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ProducerInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller**:

Public Member Functions

- **ProducerInfoMarshaller** ()
- virtual **~ProducerInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.545.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 2482). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.545.2 Constructor & Destructor Documentation

6.545.2.1 activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller::ProducerInfoMarshaller () [inline]

6.545.2.2 virtual
activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller::~~ProducerInfoMarshaller () [inline, virtual]

6.545.3 Member Function Documentation

6.545.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1331).

6.545.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1336).

6.545.3.3 virtual void activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 624).

6.545.3.4 virtual void activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 625).

6.545.3.5 virtual int activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 626).

```
6.545.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 628).

```
6.545.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 629).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ProducerInfoMarshaller.h`

6.546 **activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller** Class Reference

Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 2486).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ProducerInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller**:

Public Member Functions

- **ProducerInfoMarshaller** ()
- virtual **~ProducerInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.546.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 2486). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.546.2 Constructor & Destructor Documentation

6.546.2.1 activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller::ProducerInfoMarshaller () [inline]

6.546.2.2 virtual
activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller::~~ProducerInfoMarshaller () [inline, virtual]

6.546.3 Member Function Documentation

6.546.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1331).

6.546.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1336).

6.546.3.3 virtual void activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 618).

6.546.3.4 virtual void activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 619).

6.546.3.5 virtual int activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 620).

```
6.546.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 621).

```
6.546.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 622).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v4/**ProducerInfoMarshaller.h**

6.547 activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 2490).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ProducerInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller**:

Public Member Functions

- **ProducerInfoMarshaller** ()
- virtual **~ProducerInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.547.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 2490). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.547.2 Constructor & Destructor Documentation

6.547.2.1 activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller::ProducerInfoMarshaller () [inline]

6.547.2.2 virtual
activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller::~~ProducerInfoMarshaller () [inline, virtual]

6.547.3 Member Function Documentation

6.547.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1331).

6.547.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1336).

6.547.3.3 virtual void activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 611).

6.547.3.4 virtual void activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 612).

6.547.3.5 virtual int activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 613).

```
6.547.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 614).

```
6.547.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 615).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/ProducerInfoMarshaller.h`

6.548 activemq::state::ProducerState Class Reference

```
#include <src/main/activemq/state/ProducerState.h>
```

Public Member Functions

- **ProducerState** (const **Pointer**< **ProducerInfo** > &info)
- virtual **~ProducerState** ()
- **std::string toString** () const
- const **Pointer**< **ProducerInfo** > & **getInfo** () const

6.548.1 Constructor & Destructor Documentation

6.548.1.1 **activemq::state::ProducerState::ProducerState** (const **Pointer**< **ProducerInfo** > & *info*)

6.548.1.2 virtual **activemq::state::ProducerState::~~ProducerState** () [virtual]

6.548.2 Member Function Documentation

6.548.2.1 const **Pointer**<**ProducerInfo**>& **activemq::state::ProducerState::getInfo** () const [inline]

6.548.2.2 **std::string** **activemq::state::ProducerState::toString** () const

The documentation for this class was generated from the following file:

- `src/main/activemq/state/ProducerState.h`

6.549 decaf::util::Properties Class Reference

Java-like properties class for mapping string names to string values.

```
#include <src/main/decaf/util/Properties.h>
```

Public Member Functions

- **Properties** ()

- **Properties** (const **Properties** &src)
- virtual ~**Properties** ()
- **Properties** & **operator=** (const **Properties** &src)
Assignment Operator.
- bool **isEmpty** () const
Returns true if the properties object is empty.
- std::size_t **size** () const
- const char * **getProperty** (const std::string &name) const
Looks up the value for the given property.
- std::string **getProperty** (const std::string &name, const std::string &defaultValue) const
Looks up the value for the given property.
- void **setProperty** (const std::string &name, const std::string &value)
Sets the value for a given property.
- bool **hasProperty** (const std::string &name) const
Check to see if the Property exists in the set.
- void **remove** (const std::string &name)
Removes the property with the given name.
- std::vector< std::pair< std::string, std::string > > **toArray** () const
Method that serializes the contents of the property map to an array.
- void **copy** (const **Properties** &source)
*Copies the contents of the given properties object to this one, if the given **Properties** (p. 2494) instance in NULL then this **List** (p. 1865) is not modified.*
- **Properties** * **clone** () const
Clones this object.
- void **clear** ()
Clears all properties from the map.
- bool **equals** (const **Properties** &source) const
*Test whether two **Properties** (p. 2494) objects are equivalent.*
- std::string **toString** () const
*Formats the contents of the **Properties** (p. 2494) Object into a string that can be logged, etc.*
- void **load** (decaf::io::InputStream *stream) throw (decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::NullPointerException)
Reads a property list (key and element pairs) from the input byte stream.
- void **load** (decaf::io::Reader *reader) throw (decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::NullPointerException)

Reads a property list (key and element pairs) from the input character stream in a simple line-oriented format.

- void **store** (decaf::io::OutputStream *out, const std::string &comment) throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException)

*Writes this property list (key and element pairs) in this **Properties** (p. 2494) table to the output stream in a format suitable for loading into a **Properties** (p. 2494) table using the load(InputStream) method.*

- void **store** (decaf::io::Writer *writer, const std::string &comments) throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException)

*Writes this property list (key and element pairs) in this **Properties** (p. 2494) table to the output character stream in a format that can be read by the load(Reader) method.*

Protected Attributes

- std::auto_ptr< **Properties** > **defaults**

Default list used to answer for any keys not found in the properties list, can be filled in by another implementation of this class.

6.549.1 Detailed Description

Java-like properties class for mapping string names to string values. The **Properties** (p. 2494) list contains a key value pair of properties that can be loaded and stored to a stream. Each **Properties** (p. 2494) instance can contain an internal **Properties** (p. 2494) list that contains default values for keys not found in the **Properties** (p. 2494) **List** (p. 1865).

The **Properties** (p. 2494) list if a Thread Safe class, it can be shared amongst objects in multiple threads without the need for additional synchronization.

Since

1.0

6.549.2 Constructor & Destructor Documentation

6.549.2.1 decaf::util::Properties::Properties ()

6.549.2.2 decaf::util::Properties::Properties (const Properties & src)

6.549.2.3 virtual decaf::util::Properties::~~Properties () [virtual]

6.549.3 Member Function Documentation

6.549.3.1 void decaf::util::Properties::clear ()

Clears all properties from the map.

6.549.3.2 Properties* decaf::util::Properties::clone () const

Clones this object.

Returns

a replica of this object.

6.549.3.3 void decaf::util::Properties::copy (const Properties & source)

Copies the contents of the given properties object to this one, if the given **Properties** (p. 2494) instance is NULL then this **List** (p. 1865) is not modified.

Parameters

source The source properties object.

6.549.3.4 bool decaf::util::Properties::equals (const Properties & source) const

Test whether two **Properties** (p. 2494) objects are equivalent.

Two **Properties** (p. 2494) Objects are considered equivalent when they each contain the same number of elements and each key / value pair contained within the two are equal.

This comparison does not check the contents of the Defaults instance.

Parameters

source The **Properties** (p. 2494) object to compare this instance to.

Returns

true if the contents of the two **Properties** (p. 2494) objects are the same.

6.549.3.5 const char* decaf::util::Properties::getProperty (const std::string & name) const

Looks up the value for the given property.

Parameters

name The name of the property to be looked up.

Returns

the value of the property with the given name, if it exists. If it does not exist, returns NULL.

Referenced by decaf::util::logging::LogManager::getProperty().

6.549.3.6 `std::string decaf::util::Properties::getProperty (const std::string & name, const std::string & defaultValue) const`

Looks up the value for the given property.

Parameters

name the name of the property to be looked up.

defaultValue The value to be returned if the given property does not exist.

Returns

The value of the property specified by *name*, if it exists, otherwise the *defaultValue*.

6.549.3.7 `bool decaf::util::Properties::hasProperty (const std::string & name) const`

Check to see if the Property exists in the set.

Parameters

name - property name to check for in this properties set.

Returns

true if property exists, false otherwise.

6.549.3.8 `bool decaf::util::Properties::isEmpty () const`

Returns true if the properties object is empty.

Returns

true if empty

6.549.3.9 `void decaf::util::Properties::load (decaf::io::Reader * reader) throw (decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::NullPointerException)`

Reads a property list (key and element pairs) from the input character stream in a simple line-oriented format.

Properties (p. 2494) are processed in terms of lines. There are two kinds of line, natural lines and logical lines. A natural line is defined as a line of characters that is terminated either by a set of line terminator characters (

or or

) or by the end of the stream. A natural line may be either a blank line, a comment line, or hold all or some of a key-element pair. A logical line holds all the data of a key-element pair, which may be spread out across several adjacent natural lines by escaping the line terminator sequence with a backslash character \. Note that a comment line cannot be extended in this manner; every

natural line that is a comment must have its own comment indicator, as described below. Lines are read from input until the end of the stream is reached.

A natural line that contains only white space characters is considered blank and is ignored. A comment line has an ASCII '#' or '!' as its first non-white space character; comment lines are also ignored and do not encode key-element information. In addition to line terminators, this format considers the characters space (' '), tab (""), and form feed ("") to be white space.

If a logical line is spread across several natural lines, the backslash escaping the line terminator sequence, the line terminator sequence, and any white space at the start of the following line have no effect on the key or element values. The remainder of the discussion of key and element parsing (when loading) will assume all the characters constituting the key and element appear on a single natural line after line continuation characters have been removed. Note that it is not sufficient to only examine the character preceding a line terminator sequence to decide if the line terminator is escaped; there must be an odd number of contiguous backslashes for the line terminator to be escaped. Since the input is processed from left to right, a non-zero even number of $2n$ contiguous backslashes before a line terminator (or elsewhere) encodes n backslashes after escape processing.

The key contains all of the characters in the line starting with the first non-white space character and up to, but not including, the first unescaped '=', ':', or white space character other than a line terminator. All of these key termination characters may be included in the key by escaping them with a preceding backslash character; for example,

```
\:\=
```

would be the two-character key ":\=". Line terminator characters can be included using and

escape sequences. Any white space after the key is skipped; if the first non-white space character after the key is '=', ':', then it is ignored and any white space characters after it are also skipped. All remaining characters on the line become part of the associated element string; if there are no remaining characters, the element is the empty string "". Once the raw character sequences constituting the key and element are identified, escape processing is performed as described above.

As an example, each of the following three lines specifies the key "Truth" and the associated element value "Beauty":

```
Truth = Beauty Truth:Beauty Truth :Beauty
```

As another example, the following three lines specify a single property:

```
fruits apple, banana, pear, \ cantaloupe, watermelon, \ kiwi, mango
```

The key is "fruits" and the associated element is: "apple, banana, pear, cantaloupe, watermelon, kiwi, mango"

Note that a space appears before each \ so that a space will appear after each comma in the final result; the \, line terminator, and leading white space on the continuation line are merely discarded and are not replaced by one or more other characters.

As a third example, the line:

```
cheeses
```

specifies that the key is "cheeses" and the associated element is the empty string "".

Characters in keys and elements can be represented in escape sequences similar to those used for character and string literals (see §3.3 and §3.10.6 of the Java Language Specification). The differences from the character escape sequences and Unicode escapes used for characters and strings are:

- Octal escapes are not recognized.
- The character sequence **does** not represent a backspace character.

- The method does not treat a backslash character, `\`, before a non-valid escape character as an error; the backslash is silently dropped. For example, in a C++ string the sequence `"\z"` would cause a compile time error. In contrast, this method silently drops the backslash. Therefore, this method treats the two character sequence `"\b"` as equivalent to the single character `'b'`.
- Escapes are not necessary for single and double quotes; however, by the rule above, single and double quote characters preceded by a backslash still yield single and double quote characters, respectively.

This method does not close the Reader upon its return.

Parameters

reader The Reader that provides an character stream as input.

Exceptions

IOException if there is an error while reading from the stream.

IllegalArgumentException if malformed data is found while reading the properties.

NullPointerException if the passed stream is Null.

```
6.549.3.10 void decaf::util::Properties::load ( decaf::io::InputStream
        * stream ) throw ( decaf::io::IOException,
        decaf::lang::exceptions::IllegalArgumentException,
        decaf::lang::exceptions::NullPointerException )
```

Reads a property list (key and element pairs) from the input byte stream.

The input stream is in a simple line-oriented format as specified in `load(Reader)` and is assumed to use the ISO 8859-1 character encoding.

This method does not close the stream upon its return.

Parameters

stream The stream to read the properties data from.

Exceptions

IOException if there is an error while reading from the stream.

IllegalArgumentException if malformed data is found while reading the properties.

NullPointerException if the passed stream is Null.

```
6.549.3.11 Properties& decaf::util::Properties::operator= ( const Properties &
        src )
```

Assignment Operator.

Parameters

src The **Properties** (p. 2494) list to copy to this **List** (p. 1865).

Returns

a reference to this **List** (p. 1865) for use in chaining.

6.549.3.12 void decaf::util::Properties::remove (const std::string & *name*)

Removes the property with the given name.

Parameters

name the name of the property to remove.

6.549.3.13 void decaf::util::Properties::setProperty (const std::string & *name*,
const std::string & *value*)

Sets the value for a given property.

If the property already exists, overwrites the value.

Parameters

name The name of the value to be written.

value The value to be written.

6.549.3.14 std::size_t decaf::util::Properties::size () const**Returns**

The number of **Properties** (p. 2494) in this **Properties** (p. 2494) Object.

6.549.3.15 void decaf::util::Properties::store (decaf::io::Writer * *writer*,
const std::string & *comments*) throw (decaf::io::IOException,
decaf::lang::exceptions::NullPointerException)

Writes this property list (key and element pairs) in this **Properties** (p. 2494) table to the output character stream in a format that can be read by the load(Reader) method.

Properties (p. 2494) from the defaults table of this **Properties** (p. 2494) table (if any) are not written out by this method.

If the comments argument is not empty, then an ASCII # character, the comments string, and a line separator are first written to the output stream. Thus, the comments can serve as an identifying comment. Any one of a line feed (

'), a carriage return ("), or a carriage return followed immediately by a line feed in comments is replaced by a line separator generated by the Writer and if the next character in comments is not character # or character ! then an ASCII # is written out after that line separator.

Next, a comment line is always written, consisting of an ASCII # character, the current date and time (as if produced by the toString method of **Date** (p. 1377) for the current time), and a line separator as generated by the Writer.

Then every entry in this **Properties** (p. 2494) table is written out, one per line. For each entry the key string is written, then an ASCII =, then the associated element string. For the key, all space characters are written with a preceding \ character. For the element, leading space characters, but not embedded or trailing space characters, are written with a preceding \ character. The key and element characters #, !, =, and : are written with a preceding backslash to ensure that they are properly loaded.

After the entries have been written, the output stream is flushed. The output stream remains open after this method returns.

Parameters

- writer* The Writer instance to use to output the properties.
- comments* A description of these properties that is written before writing the properties.

Exceptions

- IOException* if there is an error while writing from the stream.
- NullPointerException* if the passed stream is Null.

6.549.3.16 `void decaf::util::Properties::store (decaf::io::OutputStream * out,
const std::string & comment) throw (decaf::io::IOException,
decaf::lang::exceptions::NullPointerException)`

Writes this property list (key and element pairs) in this **Properties** (p.2494) table to the output stream in a format suitable for loading into a **Properties** (p.2494) table using the load(InputStream) method.

Properties (p.2494) from the defaults table of this **Properties** (p.2494) table (if any) are not written out by this method.

This method outputs the comments, properties keys and values in the same format as specified in store(Writer), with the following differences:

- The stream is written using the ISO 8859-1 character encoding.
- Characters not in Latin-1 in the comments are written as for their appropriate unicode hexadecimal value xxxx.
- Characters less than and characters greater than in property keys or values are written as for the appropriate hexadecimal value xxxx.

After the entries have been written, the output stream is flushed. The output stream remains open after this method returns.

Parameters

- out* The OutputStream instance to write the properties to.
- comment* A description of these properties that is written to the output stream.

Exceptions

- IOException* if there is an error while writing from the stream.
- NullPointerException* if the passed stream is Null.

6.549.3.17 `std::vector< std::pair< std::string, std::string > >
decaf::util::Properties::toArray () const`

Method that serializes the contents of the property map to an array.

Returns

- list of pairs where the first is the name and the second is the value.

6.549.3.18 `std::string decaf::util::Properties::toString () const`

Formats the contents of the **Properties** (p. 2494) Object into a string that can be logged, etc.

Returns

string value of this object.

6.549.4 Field Documentation

6.549.4.1 `std::auto_ptr<Properties> decaf::util::Properties::defaults` [protected]

Default list used to answer for any keys not found in the properties list, can be filled in by another implementation of this class.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/Properties.h`

6.550 `decaf::util::logging::PropertiesChangeListener` Class Reference

Defines the interface that classes can use to listen for change events on **Properties** (p. 2494).

```
#include <src/main/decaf/util/logging/PropertiesChangeListener.h>
```

Public Member Functions

- `virtual ~PropertiesChangeListener ()`
- `virtual void onPropertyChanged (const std::string &name, const std::string &oldValue, const std::string &newValue)=0`

Change Event, called when a property is changed.

6.550.1 Detailed Description

Defines the interface that classes can use to listen for change events on **Properties** (p. 2494).

6.550.2 Constructor & Destructor Documentation

6.550.2.1 virtual
 decaf::util::logging::PropertiesChangeListener::~~PropertiesChangeListener
 () [inline, virtual]

6.550.3 Member Function Documentation

6.550.3.1 virtual void de-
 caf::util::logging::PropertiesChangeListener::onPropertyChanged (const
 std::string & *name*, const std::string & *oldValue*, const std::string &
newValue) [pure virtual]

Change Event, called when a property is changed.

Parameters

name - Name of the Property
oldValue - Old Value of the Property
newValue - New Value of the Property

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/PropertiesChangeListener.h

6.551 decaf::net::ProtocolException Class Reference

```
#include <src/main/decaf/net/ProtocolException.h>
```

Inheritance diagram for decaf::net::ProtocolException:

Public Member Functions

- **ProtocolException** () throw ()
Default Constructor.
- **ProtocolException** (const Exception &ex) throw ()
Conversion Constructor from some other Exception.
- **ProtocolException** (const **ProtocolException** &ex) throw ()
Copy Constructor.
- **ProtocolException** (const char *file, const int lineNumber, const std::exception *cause,
 const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **ProtocolException** (const std::exception *cause) throw ()
Constructor.

- **ProtocolException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **ProtocolException * clone** () const
Clones this exception.
- virtual ~**ProtocolException** () throw ()

6.551.1 Constructor & Destructor Documentation

6.551.1.1 decaf::net::ProtocolException::ProtocolException () throw () [inline]

Default Constructor.

6.551.1.2 decaf::net::ProtocolException::ProtocolException (const Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

Parameters

ex An exception that should become this type of Exception

References decaf::lang::Exception::Exception().

6.551.1.3 decaf::net::ProtocolException::ProtocolException (const ProtocolException & ex) throw () [inline]

Copy Constructor.

Parameters

ex An exception that should become this type of Exception

References decaf::lang::Exception::Exception().

6.551.1.4 decaf::net::ProtocolException::ProtocolException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.551.1.5 `decaf::net::ProtocolException::ProtocolException (const std::exception * cause) throw () [inline]`

Constructor.

Parameters

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.551.1.6 `decaf::net::ProtocolException::ProtocolException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.551.1.7 `virtual decaf::net::ProtocolException::~~ProtocolException () throw () [inline, virtual]`

6.551.2 Member Function Documentation

6.551.2.1 `virtual ProtocolException* decaf::net::ProtocolException::clone () const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

a new Exception instance that is a copy of this Exception object.

Reimplemented from `decaf::io::IOException` (p. 1709).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/ProtocolException.h`

6.552 decaf::security::PublicKey Class Reference

A public key.

```
#include <src/main/decaf/security/PublicKey.h>
```

Inheritance diagram for decaf::security::PublicKey:

Public Member Functions

- virtual `~PublicKey ()`

6.552.1 Detailed Description

A public key. This interface contains no methods or constants. It merely serves to group (and provide type safety for) all public key interfaces.

6.552.2 Constructor & Destructor Documentation

6.552.2.1 `virtual decaf::security::PublicKey::~~PublicKey () [inline, virtual]`

The documentation for this class was generated from the following file:

- `src/main/decaf/security/PublicKey.h`

6.553 decaf::util::Queue< E > Class Template Reference

A kind of collection provides advanced operations than other basic collections, such as insertion, extraction, and inspection.

```
#include <src/main/decaf/util/Queue.h>
```

Inheritance diagram for decaf::util::Queue< E >:

Public Member Functions

- virtual `~Queue ()`
- virtual bool **offer** (const E &value)=0 throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException)

Inserts the specified element into the queue provided that the condition allows such an operation.

- virtual bool **poll** (E &result)=0

Gets and removes the element in the head of the queue.

- virtual E **remove** ()=0 throw (decaf::lang::exceptions::NoSuchElementException)

Gets and removes the element in the head of the queue.

- virtual bool **peek** (E &result) const =0

Gets but not removes the element in the head of the queue.

- virtual E **element** () const =0 throw (decaf::lang::exceptions::NoSuchElementException)

Gets but not removes the element in the head of the queue.

6.553.1 Detailed Description

template<typename E> class decaf::util::Queue< E >

A kind of collection provides advanced operations than other basic collections, such as insertion, extraction, and inspection. Generally, a queue orders its elements by means of first-in-first-out. While priority queue orders its elements according to a comparator specified or the elements' natural order. Furthermore, a stack orders its elements last-in-first out.

Queue (p. 2507) does not provide blocking queue methods, which will block until the operation of the method is allowed. BlockingQueue interface defines such methods.

Unlike the Java **Queue** (p. 2507) interface the methods of this class cannot return null to indicate that a **Queue** (p. 2507) is empty since null has no meaning for elements such as classes, structs and primitive types and cannot be used in a meaningful way to check for an empty queue. Methods that would have returned null in the Java **Queue** (p. 2507) interface have been altered to return a boolean value indicating if the operation succeeded and take single argument that is a reference to the location where the returned value is to be assigned. This implies that elements in the **Queue** (p. 2507) must be *assignable* in order to utilize these methods.

Since

1.0

6.553.2 Constructor & Destructor Documentation

6.553.2.1 **template<typename E > virtual decaf::util::Queue< E >::~~Queue ()**
[inline, virtual]

6.553.3 Member Function Documentation

6.553.3.1 **template<typename E > virtual E decaf::util::Queue< E >::element ()**
const throw (decaf::lang::exceptions::NoSuchElementException) [pure virtual]

Gets but not removes the element in the head of the queue.

Throws a NoSuchElementException if there is no element in the queue.

Returns

the element in the head of the queue.

Exceptions

NoSuchElementException if there is no element in the queue.

Implemented in **decaf::util::AbstractQueue< E >** (p. 136).

6.553.3.2 `template<typename E > virtual bool decaf::util::Queue< E >::offer (const E & value) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException) [pure virtual]`

Inserts the specified element into the queue provided that the condition allows such an operation.

The method is generally preferable to the `collection.add(E)`, since the latter might throw an exception if the operation fails.

Parameters

value the specified element to insert into the queue.

Returns

true if the operation succeeds and false if it fails.

Exceptions

NullPointerException if the **Queue** (p. 2507) implementation does not allow Null values to be inserted into the **Queue** (p. 2507).

IllegalArgumentException if some property of the specified element prevents it from being added to this queue

Implemented in **decaf::util::PriorityQueue< E >** (p. 2417).

Referenced by `decaf::util::AbstractQueue< E >::add()`.

6.553.3.3 `template<typename E > virtual bool decaf::util::Queue< E >::peek (E & result) const [pure virtual]`

Gets but not removes the element in the head of the queue.

The result if successful is assigned to the result parameter.

Parameters

result Reference to an instance of the contained type to assigned the removed value to.

Returns

true if the element at the head of the queue was removed and assigned to the result parameter.

Implemented in **decaf::util::PriorityQueue< E >** (p. 2418).

Referenced by `decaf::util::AbstractQueue< E >::element()`.

6.553.3.4 `template<typename E > virtual bool decaf::util::Queue< E >::poll (E & result) [pure virtual]`

Gets and removes the element in the head of the queue.

If the operation succeeds the value of the element at the head of the **Queue** (p. 2507) is assigned to the result parameter and the method returns true. If the operation fails the method returns false and the value of the result parameter is undefined.

Parameters

result Reference to an instance of the contained type to assigned the removed value to.

Returns

true if the element at the head of the queue was removed and assigned to the result parameter.

Implemented in **decaf::util::PriorityQueue< E >** (p. 2418).

Referenced by **decaf::util::AbstractQueue< E >::clear()**, and **decaf::util::AbstractQueue< E >::remove()**.

6.553.3.5 `template<typename E > virtual E decaf::util::Queue< E >::remove ()
throw (decaf::lang::exceptions::NoSuchElementException) [pure
virtual]`

Gets and removes the element in the head of the queue.

Throws a **NoSuchElementException** if there is no element in the queue.

Returns

the element in the head of the queue.

Exceptions

NoSuchElementException if there is no element in the queue.

Implemented in **decaf::util::AbstractQueue< E >** (p. 136), and **decaf::util::PriorityQueue< E >** (p. 2419).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/Queue.h`

6.554 cms::Queue Class Reference

An interface encapsulating a provider-specific queue name.

```
#include <src/main/cms/Queue.h>
```

Inheritance diagram for cms::Queue:

Public Member Functions

- `virtual ~Queue ()`
- `virtual std::string getQueueName () const =0 throw (CMSException)`

Gets the name of this queue.

6.554.1 Detailed Description

An interface encapsulating a provider-specific queue name. Messages sent to a **Queue** (p. 2510) are sent to a Single Subscriber on that **Queue** (p. 2510) **Destination** (p. 1387). This allows for Queues to be used as load balances implementing a SEDA based architecture. The length of time that a Provider will store a **Message** (p. 2036) in a **Queue** (p. 2510) is not defined by the CMS API, consult your Provider documentation for this information.

Since

1.0

6.554.2 Constructor & Destructor Documentation

6.554.2.1 `virtual cms::Queue::~Queue () [inline, virtual]`

6.554.3 Member Function Documentation

6.554.3.1 `virtual std::string cms::Queue::getQueueName () const throw (CMSEException) [pure virtual]`

Gets the name of this queue.

Returns

The queue name.

Exceptions

CMSEException (p. 960) - If an internal error occurs.

Implemented in **activemq::commands::ActiveMQQueue** (p. 382).

The documentation for this class was generated from the following file:

- `src/main/cms/Queue.h`

6.555 cms::QueueBrowser Class Reference

This class implements in interface for browsing the messages in a **Queue** (p. 2510) without removing them.

```
#include <src/main/cms/QueueBrowser.h>
```

Inheritance diagram for cms::QueueBrowser:

Public Member Functions

- `virtual ~QueueBrowser ()`
- `virtual const Queue * getQueue () const =0 throw (cms::CMSEException)`
- `virtual std::string getMessageSelector () const =0 throw (cms::CMSEException)`

- virtual std::vector< const cms::Message * > **getEnumeration** () const =0 throw (cms::CMSEException)

Gets an enumeration for browsing the current queue messages in the order they would be received.

6.555.1 Detailed Description

This class implements in interface for browsing the messages in a **Queue** (p.2510) without removing them. The **getEnumeration** method of this class returns a static snapshot of the **Queue** (p.2510) at the time the method is called. Since new Message's can be arriving and old Message's could expire the client should periodically refresh its view by calling **getEnumeration** again.

Since

1.1

6.555.2 Constructor & Destructor Documentation

6.555.2.1 virtual cms::QueueBrowser::~~QueueBrowser () [inline, virtual]

6.555.3 Member Function Documentation

6.555.3.1 virtual std::vector<const cms::Message*>
cms::QueueBrowser::getEnumeration () const throw (cms::CMSEException) [pure virtual]

Gets an enumeration for browsing the current queue messages in the order they would be received.

The enumeration returned is a static view of the **Queue** (p.2510) and is not updated as new Messages arrive, the client should refresh its enumeration by calling this method again.

Returns

an STL vector for browsing the messages.

Exceptions

CMSEException (p. 960) if an internal error occurs.

6.555.3.2 virtual std::string cms::QueueBrowser::getMessageSelector () const
throw (cms::CMSEException) [pure virtual]

Returns

the MessageSelector that is used on when this browser was created or empty string if no selector was present.

Exceptions

CMSEException (p. 960) if an internal error occurs.

6.555.3.3 `virtual const Queue* cms::QueueBrowser::getQueue () const throw (cms::CMSEException)` [pure virtual]

Returns

the **Queue** (p. 2510) that this browser is listening on.

Exceptions

CMSEException (p. 960) if an internal error occurs.

The documentation for this class was generated from the following file:

- `src/main/cms/QueueBrowser.h`

6.556 decaf::util::Random Class Reference

Random (p. 2513) Value Generator which is used to generate a stream of pseudorandom numbers.

`#include <src/main/decaf/util/Random.h>`

Public Member Functions

- **Random** ()
Construct a random generator with the current time of day in milliseconds as the initial state.
- **Random** (unsigned long long seed)
*Construct a random generator with the given **seed** as the initial state.*
- `bool nextBoolean` ()
Answers the next pseudo-random, uniformly distributed boolean value generated by this generator.
- `void nextBytes` (std::vector< unsigned char > &buf)
Modifies the byte array by a random sequence of bytes generated by this random number generator.
- `double nextDouble` ()
Generates a normally distributed random double number between 0.0 inclusively and 1.0 exclusively.
- `float nextFloat` ()
Generates a normally distributed random float number between 0.0 inclusively and 1.0 exclusively.
- `double nextGaussian` ()
*Pseudo-randomly generates (approximately) a normally distributed **double** value with mean 0.0 and a standard deviation value of 1.0 using the polar method of G.*
- `int nextInt` ()
*Generates a uniformly distributed 32-bit **int** value from the this random number sequence.*
- `int nextInt` (int n) throw (lang::exceptions::IllegalArgumentException)

Returns to the caller a new pseudo-random integer value which is uniformly distributed between 0 (inclusively) and the value of `n` (exclusively).

- long long **nextLong** ()

Generates a uniformly distributed 64-bit `int` value from the this random number sequence.

- void **setSeed** (unsigned long long seed)

*Modifies the seed using linear congruential formula presented in *The Art of Computer Programming*, Volume 2, Section 3.2.1.*

Protected Member Functions

- virtual int **next** (int bits)

Answers a pseudo-random uniformly distributed `int` value of the number of bits specified by the argument `bits` as described by Donald E.

6.556.1 Detailed Description

Random (p. 2513) Value Generator which is used to generate a stream of pseudorandom numbers. The algorithms implemented by class **Random** (p. 2513) use a protected utility method that on each invocation can supply up to 32 pseudorandomly generated bits.

Since

1.0

6.556.2 Constructor & Destructor Documentation

6.556.2.1 decaf::util::Random::Random ()

Construct a random generator with the current time of day in milliseconds as the initial state.

See also

setSeed (p. 2517)

6.556.2.2 decaf::util::Random::Random (unsigned long long seed)

Construct a random generator with the given **seed** as the initial state.

Parameters

seed the seed that will determine the initial state of this random number generator

See also

setSeed (p. 2517)

6.556.3 Member Function Documentation

6.556.3.1 `virtual int decaf::util::Random::next (int bits)` [protected, virtual]

Answers a pseudo-random uniformly distributed `int` value of the number of bits specified by the argument `bits` as described by Donald E.

Knuth in *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*, section 3.2.1.

Returns

`int` a pseudo-random generated `int` number

Parameters

`bits` number of bits of the returned value

See also

`nextBytes` (p. 2515)
`nextDouble` (p. 2515)
`nextFloat` (p. 2516)
`nextInt()` (p. 2517)
`nextInt(int)` (p. 2516)
`nextGaussian` (p. 2516)
`nextLong` (p. 2517)

6.556.3.2 `bool decaf::util::Random::nextBoolean ()`

Answers the next pseudo-random, uniformly distributed boolean value generated by this generator.

Returns

`boolean` a pseudo-random, uniformly distributed boolean value

6.556.3.3 `void decaf::util::Random::nextBytes (std::vector< unsigned char > & buf)`

Modifies the byte array by a random sequence of bytes generated by this random number generator.

Parameters

`buf` non-null array to contain the new random bytes

See also

`next` (p. 2515)

6.556.3.4 `double decaf::util::Random::nextDouble ()`

Generates a normally distributed random double number between 0.0 inclusively and 1.0 exclusively.

Returns

double

See also

nextFloat (p. 2516)

6.556.3.5 float decaf::util::Random::nextFloat ()

Generates a normally distributed random float number between 0.0 inclusively and 1.0 exclusively.

Returns

float a random float number between 0.0 and 1.0

See also

nextDouble (p. 2515)

6.556.3.6 double decaf::util::Random::nextGaussian ()

Pseudo-randomly generates (approximately) a normally distributed **double** value with mean 0.0 and a standard deviation value of 1.0 using the *polar method* of G.

E. P. Box, M. E. Muller, and G. Marsaglia, as described by Donald E. Knuth in *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*, section 3.4.1, subsection C, algorithm P

Returns

double

See also

nextDouble (p. 2515)

6.556.3.7 int decaf::util::Random::nextInt (int *n*) throw (lang::exceptions::IllegalArgumentException)

Returns to the caller a new pseudo-random integer value which is uniformly distributed between 0 (inclusively) and the value of **n** (exclusively).

Returns

int

Parameters

n int

Exceptions

IllegalArgumentException

6.556.3.8 `int decaf::util::Random::nextInt ()`

Generates a uniformly distributed 32-bit `int` value from the this random number sequence.

Returns

`int` uniformly distributed `int` value

See also

`next` (p. 2515)
`nextLong` (p. 2517)

6.556.3.9 `long long decaf::util::Random::nextLong ()`

Generates a uniformly distributed 64-bit `int` value from the this random number sequence.

Returns

64-bit `int` random number

See also

`next` (p. 2515)
`nextInt()` (p. 2517)
`nextInt(int)` (p. 2516)

6.556.3.10 `void decaf::util::Random::setSeed (unsigned long long seed)`

Modifies the seed using linear congruential formula presented in *The Art of Computer Programming, Volume 2*, Section 3.2.1.

Parameters

seed the seed that alters the state of the random number generator

See also

`next` (p. 2515)
`Random()` (p. 2514)
`Random(long)`

The documentation for this class was generated from the following file:

- `src/main/decaf/util/Random.h`

6.557 `activemq::transport::inactivity::ReadChecker` Class Reference

Runnable class that is used by the {.

```
#include <src/main/activemq/transport/inactivity/ReadChecker.h>
```

Inheritance diagram for `activemq::transport::inactivity::ReadChecker`:

Public Member Functions

- **ReadChecker** (**InactivityMonitor** *parent)
- virtual **~ReadChecker** ()
- virtual void **run** ()

Run method - called by the Thread class in the context of the thread.

6.557.1 Detailed Description

Runnable class that is used by the {.

See also

InactivityMonitor (p. 1624)} class the check for timeouts related to **transport** (p. 74) reads.

Since

3.1

6.557.2 Constructor & Destructor Documentation

6.557.2.1 **activemq::transport::inactivity::ReadChecker::ReadChecker** (**InactivityMonitor** * *parent*)

6.557.2.2 **virtual activemq::transport::inactivity::ReadChecker::~~ReadChecker** (
) [virtual]

6.557.3 Member Function Documentation

6.557.3.1 **virtual void activemq::transport::inactivity::ReadChecker::run** ()
 [virtual]

Run method - called by the Thread class in the context of the thread.

Implements **decaf::lang::Runnable** (p. 2642).

The documentation for this class was generated from the following file:

- **src/main/activemq/transport/inactivity/ReadChecker.h**

6.558 decaf::io::Reader Class Reference

```
#include <src/main/decaf/io/Reader.h>
```

Public Member Functions

- virtual **~Reader** ()
- virtual void **setInputStream** (**InputStream** *is)=0

Sets the target input stream.

- virtual **InputStream * getInputStream ()**=0
Gets the target input stream.
- virtual **std::size_t read (unsigned char *buffer, std::size_t count)**=0 throw (**IOException**, **lang::exceptions::NullPointerException**)
Attempts to read an array of bytes from the stream.
- virtual **unsigned char readByte ()**=0 throw (**IOException**)
Attempts to read a byte from the input stream.

6.558.1 Constructor & Destructor Documentation

6.558.1.1 **virtual decaf::io::Reader::~~Reader ()** [inline, virtual]

6.558.2 Member Function Documentation

6.558.2.1 **virtual InputStream* decaf::io::Reader::getInputStream ()** [pure virtual]

Gets the target input stream.

6.558.2.2 **virtual std::size_t decaf::io::Reader::read (unsigned char * *buffer*, std::size_t *count*)** throw (**IOException**, **lang::exceptions::NullPointerException**) [pure virtual]

Attempts to read an array of bytes from the stream.

Parameters

buffer The target byte buffer.
count The number of bytes to read.

Returns

The number of bytes read.

Exceptions

IOException (p. 1707) thrown if an error occurs.

6.558.2.3 **virtual unsigned char decaf::io::Reader::readByte ()** throw (**IOException**) [pure virtual]

Attempts to read a byte from the input stream.

Returns

The byte.

Exceptions

IOException (p. 1707) thrown if an error occurs.

6.558.2.4 virtual void decaf::io::Reader::setInputStream (InputStream * is) [pure virtual]

Sets the target input stream.

The documentation for this class was generated from the following file:

- src/main/decaf/io/Reader.h

6.559 decaf::nio::ReadOnlyBufferException Class Reference

```
#include <src/main/decaf/nio/ReadOnlyBufferException.h>
```

Inheritance diagram for decaf::nio::ReadOnlyBufferException:

Public Member Functions

- **ReadOnlyBufferException** () throw ()
Default Constructor.
- **ReadOnlyBufferException** (const lang::Exception &ex) throw ()
Copy Constructor.
- **ReadOnlyBufferException** (const ReadOnlyBufferException &ex) throw ()
Copy Constructor.
- **ReadOnlyBufferException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **ReadOnlyBufferException** (const std::exception *cause) throw ()
Constructor.
- **ReadOnlyBufferException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor.
- virtual **ReadOnlyBufferException** * clone () const
Clones this exception.
- virtual ~**ReadOnlyBufferException** () throw ()

6.559.1 Constructor & Destructor Documentation

6.559.1.1 decaf::nio::ReadOnlyBufferException::ReadOnlyBufferException () throw () [inline]

Default Constructor.

6.559.1.2 `decaf::nio::ReadOnlyBufferException::ReadOnlyBufferException (const lang::Exception & ex) throw () [inline]`

Copy Constructor.

Parameters

ex the exception to copy

6.559.1.3 `decaf::nio::ReadOnlyBufferException::ReadOnlyBufferException (const ReadOnlyBufferException & ex) throw () [inline]`

Copy Constructor.

Parameters

ex the exception to copy, which is an instance of this type

6.559.1.4 `decaf::nio::ReadOnlyBufferException::ReadOnlyBufferException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.559.1.5 `decaf::nio::ReadOnlyBufferException::ReadOnlyBufferException (const std::exception * cause) throw () [inline]`

Constructor.

Parameters

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.559.1.6 `decaf::nio::ReadOnlyBufferException::ReadOnlyBufferException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor.

Parameters

- file* The file name where exception occurs
- lineNumber* The line number where the exception occurred.
- msg* The message to report
- ... list of primitives that are formatted into the message

6.559.1.7 virtual decaf::nio::ReadOnlyBufferException::~~ReadOnlyBufferException () throw () [inline, virtual]

6.559.2 Member Function Documentation

6.559.2.1 virtual ReadOnlyBufferException* decaf::nio::ReadOnlyBufferException::clone () const [inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Reimplemented from decaf::lang::exceptions::UnsupportedOperationException (p. 3096).

The documentation for this class was generated from the following file:

- src/main/decaf/nio/ReadOnlyBufferException.h

6.560 decaf::util::concurrent::locks::ReadWriteLock Class Reference

A **ReadWriteLock** (p. 2522) maintains a pair of associated locks, one for read-only operations and one for writing.

```
#include <src/main/decaf/util/concurrent/locks/ReadWriteLock.h>
```

Public Member Functions

- virtual ~**ReadWriteLock** ()
- virtual **Lock** & **readLock** ()=0
Returns the lock used for reading.
- virtual **Lock** & **writeLock** ()=0
Returns the lock used for writing.

6.560.1 Detailed Description

A **ReadWriteLock** (p. 2522) maintains a pair of associated locks, one for read-only operations and one for writing. The read lock may be held simultaneously by multiple reader threads, so long as there are no writers. The write lock is exclusive.

All **ReadWriteLock** (p. 2522) implementations must guarantee that the memory synchronization effects of writeLock operations (as specified in the **Lock** (p. 1898) interface) also hold with respect to the associated readLock. That is, a thread successfully acquiring the read lock will see all updates made upon previous release of the write lock.

A read-write lock allows for a greater level of concurrency in accessing shared data than that permitted by a mutual exclusion lock. It exploits the fact that while only a single thread at a time (a writer thread) can modify the shared data, in many cases any number of threads can concurrently read the data (hence reader threads). In theory, the increase in concurrency permitted by the use of a read-write lock will lead to performance improvements over the use of a mutual exclusion lock. In practice this increase in concurrency will only be fully realized on a multi-processor, and then only if the access patterns for the shared data are suitable.

Whether or not a read-write lock will improve performance over the use of a mutual exclusion lock depends on the frequency that the data is read compared to being modified, the duration of the read and write operations, and the contention for the data - that is, the number of threads that will try to read or write the data at the same time. For example, a collection that is initially populated with data and thereafter infrequently modified, while being frequently searched (such as a directory of some kind) is an ideal candidate for the use of a read-write lock. However, if updates become frequent then the data spends most of its time being exclusively locked and there is little, if any increase in concurrency. Further, if the read operations are too short the overhead of the read-write lock implementation (which is inherently more complex than a mutual exclusion lock) can dominate the execution cost, particularly as many read-write lock implementations still serialize all threads through a small section of code. Ultimately, only profiling and measurement will establish whether the use of a read-write lock is suitable for your application.

Although the basic operation of a read-write lock is straight-forward, there are many policy decisions that an implementation must make, which may affect the effectiveness of the read-write lock in a given application. Examples of these policies include:

- * Determining whether to grant the read lock or the write lock, when both readers and writers are waiting, at the time that a writer releases the write lock. Writer preference is common, as writes are expected to be short and infrequent. Reader preference is less common as it can lead to lengthy delays for a write if the readers are frequent and long-lived as expected. Fair, or "in-order" implementations are also possible.
- * Determining whether readers that request the read lock while a reader is active and a writer is waiting, are granted the read lock. Preference to the reader can delay the writer indefinitely, while preference to the writer can reduce the potential for concurrency.
- * Determining whether the locks are reentrant: can a thread with the write lock reacquire it? Can it acquire a read lock while holding the write lock? Is the read lock itself reentrant?
- * Can the write lock be downgraded to a read lock without allowing an intervening writer? Can a read lock be upgraded to a write lock, in preference to other waiting readers or writers?

You should consider all of these things when evaluating the suitability of a given implementation for your application.

Since

1.0

6.560.2 Constructor & Destructor Documentation

6.560.2.1 virtual decaf::util::concurrent::locks::ReadWriteLock::~~ReadWriteLock () [inline, virtual]

6.560.3 Member Function Documentation

6.560.3.1 virtual Lock& decaf::util::concurrent::locks::ReadWriteLock::readLock () [pure virtual]

Returns the lock used for reading.

Returns

the lock used for reading.

6.560.3.2 virtual Lock& decaf::util::concurrent::locks::ReadWriteLock::writeLock () [pure virtual]

Returns the lock used for writing.

Returns

the lock used for writing.

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/locks/ReadWriteLock.h

6.561 activemq::cmsutil::CmsTemplate::ReceiveExecutor Class Reference

```
#include <src/main/activemq/cmsutil/CmsTemplate.h>
```

Inheritance diagram for activemq::cmsutil::CmsTemplate::ReceiveExecutor:

Public Member Functions

- **ReceiveExecutor** (CmsTemplate *parent, cms::Destination *destination, const std::string &selector, bool noLocal)
- virtual ~**ReceiveExecutor** ()
- virtual void **doInCms** (cms::Session *session) throw (cms::CMSException)
Execute any number of operations against the supplied CMS session.
- virtual cms::Destination * **getDestination** (cms::Session *session AMQCPP_UNUSED) throw (cms::CMSException)
- cms::Message * **getMessage** ()

Protected Attributes

- `cms::Destination * destination`
- `std::string selector`
- `bool noLocal`
- `cms::Message * message`
- `CmsTemplate * parent`

6.561.1 Constructor & Destructor Documentation

6.561.1.1 `activemq::cmsutil::CmsTemplate::ReceiveExecutor::ReceiveExecutor (CmsTemplate * parent, cms::Destination * destination, const std::string & selector, bool noLocal)` [inline]

6.561.1.2 `virtual activemq::cmsutil::CmsTemplate::ReceiveExecutor::~~ReceiveExecutor ()` [inline, virtual]

6.561.2 Member Function Documentation

6.561.2.1 `virtual void activemq::cmsutil::CmsTemplate::ReceiveExecutor::doInCms (cms::Session * session) throw (cms::CMSException)` [virtual]

Execute any number of operations against the supplied CMS session.

Parameters

session the CMS Session

Exceptions

cms::CMSException (p. 960) if thrown by CMS API methods

Implements `activemq::cmsutil::SessionCallback` (p. 2677).

- 6.561.2.2 virtual cms::Destination* activemq::cmsutil::CmsTemplate::ReceiveExecutor::getDestination (cms::Session *session *AMQCPP_UNUSED*) throw (cms::CMSException) [inline, virtual]
- 6.561.2.3 cms::Message* activemq::cmsutil::CmsTemplate::ReceiveExecutor::getMessage () [inline]
- 6.561.3 Field Documentation
- 6.561.3.1 cms::Destination* activemq::cmsutil::CmsTemplate::ReceiveExecutor::destination [protected]
- 6.561.3.2 cms::Message* activemq::cmsutil::CmsTemplate::ReceiveExecutor::message [protected]
- 6.561.3.3 bool activemq::cmsutil::CmsTemplate::ReceiveExecutor::noLocal [protected]
- 6.561.3.4 CmsTemplate* activemq::cmsutil::CmsTemplate::ReceiveExecutor::parent [protected]
- 6.561.3.5 std::string activemq::cmsutil::CmsTemplate::ReceiveExecutor::selector [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/CmsTemplate.h

6.562 decaf::util::concurrent::locks::ReentrantLock Class Reference

A reentrant mutual exclusion **Lock** (p. 1898) with extended capabilities.

```
#include <src/main/decaf/util/concurrent/locks/ReentrantLock.h>
```

Inheritance diagram for decaf::util::concurrent::locks::ReentrantLock:

Public Member Functions

- **ReentrantLock** ()
- virtual ~**ReentrantLock** ()
- virtual void **lock** () throw (decaf::lang::exceptions::RuntimeException)

Acquires the lock.

- virtual void **lockInterruptibly** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::InterruptedException)
Acquires the lock unless the current thread is interrupted.
- virtual bool **tryLock** () throw (decaf::lang::exceptions::RuntimeException)
Acquires the lock only if it is not held by another thread at the time of invocation.
- virtual bool **tryLock** (long long time, const **TimeUnit** &unit) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::InterruptedException)
Acquires the lock if it is not held by another thread within the given waiting time and the current thread has not been interrupted.
- virtual void **unlock** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)
Attempts to release this lock.
- virtual **Condition** * **newCondition** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::UnsupportedOperationException)
*Returns a **Condition** (p. 1040) instance for use with this **Lock** (p. 1898) instance.*
- int **getHoldCount** () const
Queries the number of holds on this lock by the current thread.
- bool **isHeldByCurrentThread** () const
Queries if this lock is held by the current thread.
- bool **isLocked** () const
Queries if this lock is held by any thread.
- bool **isFair** () const
Returns true if this lock has fairness set true.
- std::string **toString** () const
Returns a string identifying this lock, as well as its lock state.

6.562.1 Detailed Description

A reentrant mutual exclusion **Lock** (p. 1898) with extended capabilities. A **ReentrantLock** (p. 2526) is owned by the thread last successfully locking, but not yet unlocking it. A thread invoking lock will return, successfully acquiring the lock, when the lock is not owned by another thread. The method will return immediately if the current thread already owns the lock. This can be checked using methods **isHeldByCurrentThread**() (p. 2529), and **getHoldCount**() (p. 2528).

The constructor for this class accepts an optional fairness parameter. When set true, under contention, locks favor granting access to the longest-waiting thread. Otherwise this lock does not guarantee any particular access order. Programs using fair locks accessed by many threads may display lower overall throughput (i.e., are slower; often much slower) than those using the default setting, but have smaller variances in times to obtain locks and guarantee lack of starvation. Note however, that fairness of locks does not guarantee fairness of thread scheduling. Thus, one of many

threads using a fair lock may obtain it multiple times in succession while other active threads are not progressing and not currently holding the lock. Also note that the untimed tryLock method does not honor the fairness setting. It will succeed if the lock is available even if other threads are waiting.

It is recommended practice to always immediately follow a call to lock with a try block, most typically in a before/after construction such as:

```
class X { private:
ReentrantLock (p. 2526) lock; // ...
public:
void m() { lock.lock(); // block until condition holds
try { // ... method body } finally { lock.unlock() } } }
```

In addition to implementing the **Lock** (p. 1898) interface, this class defines methods isLocked and getLockQueueLength, as well as some associated protected access methods that may be useful for instrumentation and monitoring.

Since

1.0

6.562.2 Constructor & Destructor Documentation

6.562.2.1 decaf::util::concurrent::locks::ReentrantLock::ReentrantLock ()

6.562.2.2 virtual decaf::util::concurrent::locks::ReentrantLock::~~ReentrantLock () [virtual]

6.562.3 Member Function Documentation

6.562.3.1 int decaf::util::concurrent::locks::ReentrantLock::getHoldCount ()
const

Queries the number of holds on this lock by the current thread.

A thread has a hold on a lock for each lock action that is not matched by an unlock action.

The hold count information is typically only used for testing and debugging purposes. For example, if a certain section of code should not be entered with the lock already held then we can assert that fact:

```
class X { private:
ReentrantLock (p. 2526) lock; // ...
public:
void m() { assert( lock.getHoldCount() == 0 ); lock.lock(); try { // ... method body } catch(...)
{ lock.unlock(); } } }
```

Returns

the number of holds on this lock by the current thread, or zero if this lock is not held by the current thread

6.562.3.2 bool decaf::util::concurrent::locks::ReentrantLock::isFair () const

Returns true if this lock has fairness set true.

Returns

true if this lock has fairness set true

6.562.3.3 bool decaf::util::concurrent::locks::ReentrantLock::isHeldByCurrentThread () const

Queries if this lock is held by the current thread.

This method is typically used for debugging and testing. For example, a method that should only be called while a lock is held can assert that this is the case:

```
class X { private: ReentrantLock (p. 2526) lock = new ReentrantLock() (p. 2528); // ...
public: void m() { assert( lock.isHeldByCurrentThread() ); // ... method body } }
```

It can also be used to ensure that a reentrant lock is used in a non-reentrant manner, for example:

```
class X { private: ReentrantLock (p. 2526) lock = new ReentrantLock() (p. 2528); // ...
public: void m() { assert !lock.isHeldByCurrentThread() (p. 2529); lock.lock(); try { // ...
method body } finally { lock.unlock(); } } }
```

Returns

true if current thread holds this lock and false otherwise

6.562.3.4 bool decaf::util::concurrent::locks::ReentrantLock::isLocked () const

Queries if this lock is held by any thread.

This method is designed for use in monitoring of the system state, not for synchronization control.

Returns

true if any thread holds this lock and false otherwise

6.562.3.5 virtual void decaf::util::concurrent::locks::ReentrantLock::lock () throw (decaf::lang::exceptions::RuntimeException) [virtual]

Acquires the lock.

Acquires the lock if it is not held by another thread and returns immediately, setting the lock hold count to one.

If the current thread already holds the lock then the hold count is incremented by one and the method returns immediately.

If the lock is held by another thread then the current thread becomes disabled for thread scheduling purposes and lies dormant until the lock has been acquired, at which time the lock hold count is set to one.

Exceptions

RuntimeException if an error occurs while acquiring the lock.

Implements **decaf::util::concurrent::locks::Lock** (p. 1900).

6.562.3.6 `virtual void decaf::util::concurrent::locks::ReentrantLock::lockInterruptibly () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::InterruptedException) [virtual]`

Acquires the lock unless the current thread is interrupted.

Acquires the lock if it is not held by another thread and returns immediately, setting the lock hold count to one.

If the current thread already holds this lock then the hold count is incremented by one and the method returns immediately.

If the lock is held by another thread then the current thread becomes disabled for thread scheduling purposes and lies dormant until one of two things happens:

* The lock is acquired by the current thread; or * Some other thread interrupts the current thread.

If the lock is acquired by the current thread then the lock hold count is set to one.

If the current thread:

* has its interrupted status set on entry to this method; or * is interrupted while acquiring the lock,

then **InterruptedException** is thrown and the current thread's interrupted status is cleared.

In this implementation, as this method is an explicit interruption point, preference is given to responding to the interrupt over normal or reentrant acquisition of the lock.

Exceptions

RuntimeException if an error occurs while acquiring the lock.

InterruptedException if the current thread is interrupted while acquiring the lock (and interruption of lock acquisition is supported).

Implements **decaf::util::concurrent::locks::Lock** (p. 1900).

6.562.3.7 `virtual Condition* decaf::util::concurrent::locks::ReentrantLock::newCondition () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Returns a **Condition** (p. 1040) instance for use with this **Lock** (p. 1898) instance.

The returned **Condition** (p. 1040) instance supports the same usages as do the **Mutex** (p. 2239) Class' methods (wait, notify, and notifyAll).

* If this lock is not held when any of the **Condition** (p. 1040) waiting or signalling methods are called, then an **IllegalMonitorStateException** is thrown. * When the condition waiting methods are called the lock is released and, before they return, the lock is reacquired and the lock hold count restored to what it was when the method was called. * If a thread is interrupted while

waiting then the wait will terminate, an `InterruptedException` will be thrown, and the thread's interrupted status will be cleared. * Waiting threads are signaled in FIFO order. * The ordering of lock reacquisition for threads returning from waiting methods is the same as for threads initially acquiring the lock, which is in the default case not specified, but for fair locks favors those threads that have been waiting the longest.

Exceptions

RuntimeException if an error occurs while creating the **Condition** (p. 1040).

UnsupportedOperationException if this **Lock** (p. 1898) implementation does not support conditions

Implements **decaf::util::concurrent::locks::Lock** (p. 1901).

6.562.3.8 `std::string decaf::util::concurrent::locks::ReentrantLock::toString ()` `const`

Returns a string identifying this lock, as well as its lock state.

The state, in brackets, includes either the String "Unlocked" or the String "Locked by" followed by the name of the owning thread.

Returns

a string identifying this lock, as well as its lock state

6.562.3.9 `virtual bool decaf::util::concurrent::locks::ReentrantLock::tryLock` `(long long time, const TimeUnit & unit)` `throw (decaf::lang::exceptions::RuntimeException,` `decaf::lang::exceptions::InterruptedException) [virtual]`

Acquires the lock if it is not held by another thread within the given waiting time and the current thread has not been interrupted.

Acquires the lock if it is not held by another thread and returns immediately with the value true, setting the lock hold count to one. If this lock has been set to use a fair ordering policy then an available lock will not be acquired if any other threads are waiting for the lock. This is in contrast to the **tryLock()** (p. 2532) method. If you want a timed tryLock that does permit barging on a fair lock then combine the timed and un-timed forms together:

```
if (lock.tryLock() || lock.tryLock(timeout, unit) ) { ... }
```

If the current thread already holds this lock then the hold count is incremented by one and the method returns true.

If the lock is held by another thread then the current thread becomes disabled for thread scheduling purposes and lies dormant until one of three things happens:

* The lock is acquired by the current thread; or
* Some other thread interrupts the current thread;
or
* The specified waiting time elapses

If the lock is acquired then the value true is returned and the lock hold count is set to one.

If the current thread:

* has its interrupted status set on entry to this method; or
* is interrupted while acquiring the lock,

then InterruptedException is thrown and the current thread's interrupted status is cleared.

If the specified waiting time elapses then the value false is returned. If the time is less than or equal to zero, the method will not wait at all.

In this implementation, as this method is an explicit interruption point, preference is given to responding to the interrupt over normal or reentrant acquisition of the lock, and over reporting the elapse of the waiting time.

Parameters

time the maximum time to wait for the lock

unit the time unit of the time argument

Returns

true if the lock was acquired and false if the waiting time elapsed before the lock was acquired

Exceptions

RuntimeException if an error occurs while acquiring the lock.

InterruptedException if the current thread is interrupted while acquiring the lock (and interruption of lock acquisition is supported)

Implements **decaf::util::concurrent::locks::Lock** (p. 1901).

6.562.3.10 `virtual bool decaf::util::concurrent::locks::ReentrantLock::tryLock ()
throw (decaf::lang::exceptions::RuntimeException) [virtual]`

Acquires the lock only if it is not held by another thread at the time of invocation.

Acquires the lock if it is not held by another thread and returns immediately with the value true, setting the lock hold count to one. Even when this lock has been set to use a fair ordering policy, a call to **tryLock()** (p. 2532) will immediately acquire the lock if it is available, whether or not other threads are currently waiting for the lock. This "barging" behavior can be useful in certain circumstances, even though it breaks fairness. If you want to honor the fairness setting for this lock, then use **tryLock(0, TimeUnit.SECONDS)** (p. 3014) which is almost equivalent (it also detects interruption).

If the current thread already holds this lock then the hold count is incremented by one and the method returns true.

If the lock is held by another thread then this method will return immediately with the value false.

Returns

true if the lock was acquired and false otherwise

Exceptions

RuntimeException if an error occurs while acquiring the lock.

Implements **decaf::util::concurrent::locks::Lock** (p. 1902).

6.562.3.11 `virtual void decaf::util::concurrent::locks::ReentrantLock::unlock
() throw (decaf::lang::exceptions::RuntimeException,
decaf::lang::exceptions::IllegalMonitorStateException) [virtual]`

Attempts to release this lock.

If the current thread is the holder of this lock then the hold count is decremented. If the hold count is now zero then the lock is released. If the current thread is not the holder of this lock then `IllegalMonitorStateException` is thrown.

Exceptions

RuntimeException if an error occurs while acquiring the lock.

Implements `decaf::util::concurrent::locks::Lock` (p. 1903).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/locks/ReentrantLock.h`

6.563 decaf::util::concurrent::RejectedExecutionException Class Reference

```
#include <src/main/decaf/util/concurrent/RejectedExecutionException.h>
```

Inheritance diagram for `decaf::util::concurrent::RejectedExecutionException`:

Public Member Functions

- **RejectedExecutionException** () throw ()
Default Constructor.
- **RejectedExecutionException** (const Exception &ex) throw ()
Conversion Constructor from some other Exception.
- **RejectedExecutionException** (const **RejectedExecutionException** &ex) throw ()
Copy Constructor.
- **RejectedExecutionException** (const std::exception *cause) throw ()
Constructor.
- **RejectedExecutionException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **RejectedExecutionException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.

- virtual **RejectedExecutionException** * clone () const
Clones this exception.
- virtual ~**RejectedExecutionException** () throw ()

6.563.1 Constructor & Destructor Documentation

6.563.1.1 decaf::util::concurrent::RejectedExecutionException::RejectedExecutionException () throw () [inline]

Default Constructor.

6.563.1.2 decaf::util::concurrent::RejectedExecutionException::RejectedExecutionException (const Exception & *ex*) throw () [inline]

Conversion Constructor from some other Exception.

Parameters

ex An exception that should become this type of Exception

References decaf::lang::Exception::Exception().

6.563.1.3 decaf::util::concurrent::RejectedExecutionException::RejectedExecutionException (const RejectedExecutionException & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex - The Exception to copy in this new instance.

References decaf::lang::Exception::Exception().

6.563.1.4 decaf::util::concurrent::RejectedExecutionException::RejectedExecutionException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.563.1.5 decaf::util::concurrent::RejectedExecutionException::RejectedExecutionException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

- file* - The file name where exception occurs
- lineNumber* - The line number where the exception occurred.
- msg* - The message to report
- ... - list of primitives that are formatted into the message

6.563.1.6 `decaf::util::concurrent::RejectedExecutionException::RejectedExecutionException`
 (`const char * file`, `const int lineNumber`, `const std::exception * cause`, `const char * msg`, ...) `throw ()` [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

- file* - The file name where exception occurs
- lineNumber* - The line number where the exception occurred.
- cause* - The exception that was the cause for this one to be thrown.
- msg* - The message to report
- ... - list of primitives that are formatted into the message

6.563.1.7 `virtual`
`decaf::util::concurrent::RejectedExecutionException::~RejectedExecutionException`
 () `throw ()` [inline, virtual]

6.563.2 Member Function Documentation

6.563.2.1 `virtual RejectedExecutionException* decaf::util::concurrent::RejectedExecutionException::clone` () `const` [inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

A new instance this exception type with a copy the current state.

Reimplemented from `decaf::lang::Exception` (p. 1479).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/RejectedExecutionException.h`

6.564 decaf::util::concurrent::RejectedExecutionHandler Class Reference

A handler for tasks that cannot be executed by a **ThreadPoolExecutor** (p. ??).

#include <src/main/decaf/util/concurrent/RejectedExecutionHandler.h>

Public Member Functions

- virtual ~**RejectedExecutionHandler** ()

6.564.1 Detailed Description

A handler for tasks that cannot be executed by a **ThreadPoolExecutor** (p. ??).

Since

1.0

6.564.2 Constructor & Destructor Documentation

- 6.564.2.1 virtual
decaf::util::concurrent::RejectedExecutionHandler::~RejectedExecutionHandler
() [inline, virtual]

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**RejectedExecutionHandler.h**

6.565 activemq::commands::RemoveInfo Class Reference

#include <src/main/activemq/commands/RemoveInfo.h>

Inheritance diagram for activemq::commands::RemoveInfo:

Public Member Functions

- **RemoveInfo** ()
- virtual ~**RemoveInfo** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **RemoveInfo** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)

Copy the contents of the passed object into this object's members, overwriting any existing data.

- virtual std::string **toString** () const

*Returns a string containing the information for this **DataSet** (p. 1372) such as its type and value of its elements.*

- virtual bool **equals** (const **DataSet** *value) const

*Compares the **DataSet** (p. 1372) passed in to this one, and returns if they are equivalent.*

- virtual const **Pointer**< **DataSet** > & **getObjectId** () const
- virtual **Pointer**< **DataSet** > & **getObjectId** ()
- virtual void **setObjectId** (const **Pointer**< **DataSet** > &objectId)
- virtual long long **getLastDeliveredSequenceId** () const
- virtual void **setLastDeliveredSequenceId** (long long lastDeliveredSequenceId)
- virtual bool **isRemoveInfo** () const
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor *visitor) throw (exceptions::ActiveMQException)

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_REMOVEINFO** = 12

Protected Member Functions

- **RemoveInfo** (const **RemoveInfo** &)
- **RemoveInfo** & **operator=** (const **RemoveInfo** &)

Protected Attributes

- **Pointer**< **DataSet** > **objectId**
- long long **lastDeliveredSequenceId**

6.565.1 Constructor & Destructor Documentation

6.565.1.1 `activemq::commands::RemoveInfo::RemoveInfo (const RemoveInfo &) [inline, protected]`

6.565.1.2 `activemq::commands::RemoveInfo::RemoveInfo ()`

6.565.1.3 `virtual activemq::commands::RemoveInfo::~~RemoveInfo () [virtual]`

6.565.2 Member Function Documentation

6.565.2.1 `virtual RemoveInfo* activemq::commands::RemoveInfo::cloneDataStructure () const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1373).

6.565.2.2 `virtual void activemq::commands::RemoveInfo::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

Reimplemented from `activemq::commands::BaseCommand` (p. 598).

6.565.2.3 `virtual bool activemq::commands::RemoveInfo::equals (const DataStructure * value) const [virtual]`

Compares the `DataStructure` (p. 1372) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 598).

6.565.2.4 `virtual unsigned char activemq::commands::RemoveInfo::getDataStructureType () const [virtual]`

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataSet** (p. 1372) type copy.

Implements **activemq::commands::DataSet** (p. 1375).

6.565.2.5 virtual long long **activemq::commands::RemoveInfo::getLastDeliveredSequenceId** () const [virtual]

6.565.2.6 virtual Pointer<DataSet>& **activemq::commands::RemoveInfo::getObjectId** () [virtual]

6.565.2.7 virtual const Pointer<DataSet>& **activemq::commands::RemoveInfo::getObjectId** () const [virtual]

6.565.2.8 virtual bool **activemq::commands::RemoveInfo::isRemoveInfo** () const [inline, virtual]

Returns

an answer of true to the **isRemoveInfo()** (p. 2539) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 601).

6.565.2.9 RemoveInfo& **activemq::commands::RemoveInfo::operator=** (const RemoveInfo &) [inline, protected]

6.565.2.10 virtual void **activemq::commands::RemoveInfo::setLastDeliveredSequenceId** (long long *lastDeliveredSequenceId*) [virtual]

6.565.2.11 virtual void **activemq::commands::RemoveInfo::setObjectId** (const Pointer< DataSet > & *objectId*) [virtual]

6.565.2.12 virtual std::string **activemq::commands::RemoveInfo::toString** () const [virtual]

Returns a string containing the information for this **DataSet** (p. 1372) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 602).

6.565.2.13 `virtual Pointer<Command> activemq::commands::RemoveInfo::visit
 (activemq::state::CommandVisitor * visitor) throw (
 exceptions::ActiveMQException) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

Returns

a **Response** (p. 2611) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 995).

6.565.3 Field Documentation

6.565.3.1 `const unsigned char activemq::commands::RemoveInfo::ID_ -
 REMOVEINFO = 12 [static]`

6.565.3.2 `long long activemq::commands::RemoveInfo::lastDeliveredSequenceId
 [protected]`

6.565.3.3 `Pointer<DataStructure> activemq::commands::RemoveInfo::objectId
 [protected]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/RemoveInfo.h`

6.566 activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 2540).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/RemoveInfoMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller`:

Public Member Functions

- **RemoveInfoMarshaller** ()
- virtual **~RemoveInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.566.1 Detailed Description

Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 2540). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.566.2 Constructor & Destructor Documentation

6.566.2.1 **activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller::RemoveInfoMarshaller**
() [inline]

6.566.2.2 **virtual**
activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller::~~RemoveInfoMarshaller
() [inline, virtual]

6.566.3 Member Function Documentation

6.566.3.1 **virtual commands::DataStructure*** **activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller::createObject**
() const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1331).

6.566.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller::getDataSetType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1336).

6.566.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataSet * dataSet, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 611).

6.566.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataSet * dataSet, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 612).

6.566.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 613).

6.566.3.6 `virtual void activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 614).

6.566.3.7 `virtual void activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 615).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/RemoveInfoMarshaller.h`

6.567 activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 2544).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/RemoveInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller**:

Public Member Functions

- **RemoveInfoMarshaller** ()
- virtual **~RemoveInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.567.1 Detailed Description

Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 2544). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.567.2 Constructor & Destructor Documentation

6.567.2.1 **activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller::RemoveInfoMarshaller** () [inline]

6.567.2.2 **virtual**
activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller::~~RemoveInfoMarshaller
() [inline, virtual]

6.567.3 Member Function Documentation

6.567.3.1 **virtual commands::DataStructure*** **activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller::createObject** () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1331).

6.567.3.2 **virtual unsigned char** **activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller::getDataStructureType** () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1336).

6.567.3.3 virtual void activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 624).

6.567.3.4 virtual void activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 625).

6.567.3.5 virtual int activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 626).

```
6.567.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 628).

```
6.567.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 629).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/RemoveInfoMarshaller.h`

6.568 activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 2548).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/RemoveInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller**:

Public Member Functions

- **RemoveInfoMarshaller** ()
- virtual **~RemoveInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.568.1 Detailed Description

Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p.2548). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.568.2 Constructor & Destructor Documentation

6.568.2.1 activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller::RemoveInfoMarshaller () [inline]

6.568.2.2 virtual
activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller::~~RemoveInfoMarshaller () [inline, virtual]

6.568.3 Member Function Documentation

6.568.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1331).

6.568.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1336).

6.568.3.3 virtual void activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 631).

6.568.3.4 virtual void activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 632).

6.568.3.5 virtual int activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 633).

```
6.568.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 634).

```
6.568.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 635).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/RemoveInfoMarshaller.h`

6.569 activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 2552).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/RemoveInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller**:

Public Member Functions

- **RemoveInfoMarshaller** ()
- virtual **~RemoveInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.569.1 Detailed Description

Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p.2552). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.569.2 Constructor & Destructor Documentation

6.569.2.1 activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller::RemoveInfoMarshaller () [inline]

6.569.2.2 virtual
activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller::~~RemoveInfoMarshaller () [inline, virtual]

6.569.3 Member Function Documentation

6.569.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1331).

6.569.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1336).

6.569.3.3 virtual void activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 604).

6.569.3.4 virtual void activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 605).

6.569.3.5 virtual int activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 606).

```
6.569.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 608).

```
6.569.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 609).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**RemoveInfoMarshaller.h**

6.570 activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 2556).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/RemoveInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller**:

Public Member Functions

- **RemoveInfoMarshaller** ()
- virtual **~RemoveInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.

- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)`

Write a object instance to data output stream.

6.570.1 Detailed Description

Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p.2556). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.570.2 Constructor & Destructor Documentation

6.570.2.1 `activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller::RemoveInfoMarshaller () [inline]`

6.570.2.2 `virtual activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller::~~RemoveInfoMarshaller () [inline, virtual]`

6.570.3 Member Function Documentation

6.570.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1331).

6.570.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1336).

6.570.3.3 virtual void activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 618).

6.570.3.4 virtual void activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 619).

6.570.3.5 virtual int activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 620).

```
6.570.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 621).

```
6.570.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 622).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/RemoveInfoMarshaller.h`

6.571 activemq::commands::RemoveSubscriptionInfo Class Reference

```
#include <src/main/activemq/commands/RemoveSubscriptionInfo.h>
```

Inheritance diagram for **activemq::commands::RemoveSubscriptionInfo**:

Public Member Functions

- **RemoveSubscriptionInfo** ()
- virtual **~RemoveSubscriptionInfo** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaller share.
- virtual **RemoveSubscriptionInfo * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1372) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **ConnectionId** > & **getConnectionId** () const
- virtual **Pointer**< **ConnectionId** > & **getConnectionId** ()
- virtual void **setConnectionId** (const **Pointer**< **ConnectionId** > &connectionId)
- virtual const std::string & **getSubscriptionName** () const
- virtual std::string & **getSubscriptionName** ()
- virtual void **setSubscriptionName** (const std::string &subscriptionName)
- virtual const std::string & **getClientId** () const
- virtual std::string & **getClientId** ()
- virtual void **setClientId** (const std::string &clientId)

- virtual bool **isRemoveSubscriptionInfo** () const
- virtual **Pointer**< **Command** > **visit** (**activemq::state::CommandVisitor** *visitor) throw (exceptions::ActiveMQException)

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_REMOVESUBSCRIPTIONINFO** = 9

Protected Member Functions

- **RemoveSubscriptionInfo** (const **RemoveSubscriptionInfo** &)
- **RemoveSubscriptionInfo** & **operator=** (const **RemoveSubscriptionInfo** &)

Protected Attributes

- **Pointer**< **ConnectionId** > **connectionId**
- std::string **subscriptionName**
- std::string **clientId**

6.571.1 Constructor & Destructor Documentation

6.571.1.1 **activemq::commands::RemoveSubscriptionInfo::RemoveSubscriptionInfo** (const **RemoveSubscriptionInfo** &) [inline, protected]

6.571.1.2 **activemq::commands::RemoveSubscriptionInfo::RemoveSubscriptionInfo** ()

6.571.1.3 virtual **activemq::commands::RemoveSubscriptionInfo::~~RemoveSubscriptionInfo** () [virtual]

6.571.2 Member Function Documentation

6.571.2.1 virtual **RemoveSubscriptionInfo*** **activemq::commands::RemoveSubscriptionInfo::cloneDataStructure** () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1373).

6.571.2.2 virtual void activemq::commands::RemoveSubscriptionInfo::copyDataStructure (const DataStructure * *src*) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 598).

6.571.2.3 virtual bool activemq::commands::RemoveSubscriptionInfo::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 598).

6.571.2.4 virtual const std::string& activemq::commands::RemoveSubscriptionInfo::getClientId () const [virtual]

6.571.2.5 virtual std::string& activemq::commands::RemoveSubscriptionInfo::getClientId () [virtual]

6.571.2.6 virtual Pointer<ConnectionId>& activemq::commands::RemoveSubscriptionInfo::getConnectionId () [virtual]

6.571.2.7 virtual const Pointer<ConnectionId>& activemq::commands::RemoveSubscriptionInfo::getConnectionId () const [virtual]

6.571.2.8 virtual unsigned char activemq::commands::RemoveSubscriptionInfo::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataStructure** (p. 1372) type copy.

Implements **activemq::commands::DataStructure** (p. 1375).

- 6.571.2.9** `virtual const std::string& activemq::commands::RemoveSubscriptionInfo::getSubscriptionName () const [virtual]`
- 6.571.2.10** `virtual std::string& activemq::commands::RemoveSubscriptionInfo::getSubscriptionName () [virtual]`
- 6.571.2.11** `virtual bool activemq::commands::RemoveSubscriptionInfo::isRemoveSubscriptionInfo () const [inline, virtual]`

Returns

an answer of true to the `isRemoveSubscriptionInfo()` (p. 2563) query.

Reimplemented from `activemq::commands::BaseCommand` (p. 601).

- 6.571.2.12** `RemoveSubscriptionInfo& activemq::commands::RemoveSubscriptionInfo::operator= (const RemoveSubscriptionInfo &) [inline, protected]`
- 6.571.2.13** `virtual void activemq::commands::RemoveSubscriptionInfo::setClientId (const std::string & clientId) [virtual]`
- 6.571.2.14** `virtual void activemq::commands::RemoveSubscriptionInfo::setConnectionId (const Pointer< ConnectionId > & connectionId) [virtual]`
- 6.571.2.15** `virtual void activemq::commands::RemoveSubscriptionInfo::setSubscriptionName (const std::string & subscriptionName) [virtual]`
- 6.571.2.16** `virtual std::string activemq::commands::RemoveSubscriptionInfo::toString () const [virtual]`

Returns a string containing the information for this **DataStructure** (p. 1372) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from `activemq::commands::BaseCommand` (p. 602).

- 6.571.2.17** `virtual Pointer<Command> activemq::commands::RemoveSubscriptionInfo::visit (activemq::state::CommandVisitor * visitor) throw (exceptions::ActiveMQException) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

6.572

activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller Class Reference

2567

The command will call the proper processXXX method in the visitor.

Returns

a **Response** (p. 2611) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 995).

6.571.3 Field Documentation

6.571.3.1 `std::string activemq::commands::RemoveSubscriptionInfo::clientId`
[protected]

6.571.3.2 `Pointer<ConnectionId> activemq::commands::RemoveSubscriptionInfo::connectionId`
[protected]

6.571.3.3 `const unsigned char activemq::commands::RemoveSubscriptionInfo::ID_REMOVE_SUBSCRIPTIONINFO = 9` [static]

6.571.3.4 `std::string activemq::commands::RemoveSubscriptionInfo::subscriptionName`
[protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/RemoveSubscriptionInfo.h`

6.572 activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 2564).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/RemoveSubscriptionInfoMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller`:

Public Member Functions

- **RemoveSubscriptionInfoMarshaller** ()
- virtual **~RemoveSubscriptionInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.572.1 Detailed Description

Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 2564).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.572.2 Constructor & Destructor Documentation

6.572.2.1 activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller::RemoveSubscriptionInfoMarshaller () [inline]

6.572.2.2 virtual
 activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller::~~RemoveSubscriptionInfoMarshaller () [inline, virtual]

6.572.3 Member Function Documentation

6.572.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

6.572

activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller

Class Reference

2569

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1331).

6.572.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller::getDataStructureType() const [virtual]`

Get the Data Structure Type that identifies this Marshaller.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1336).

6.572.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller::marshal(const OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 624).

6.572.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller::unmarshal(const OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 625).

6.572.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 626).

6.572.3.6 `virtual void activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 628).

6.572.3.7 `virtual void activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 629).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/RemoveSubscriptionInfoMarshaller.h`

6.573 activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 2568).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/RemoveSubscriptionInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller**:

Public Member Functions

- **RemoveSubscriptionInfoMarshaller** ()
- virtual **~RemoveSubscriptionInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.573.1 Detailed Description

Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 2568).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.573.2 Constructor & Destructor Documentation

6.573.2.1 **activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller::RemoveSubscriptionInfoMarshaller** () [inline]

6.573.2.2 virtual **activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller::~RemoveSubscriptionInfoMarshaller** () [inline, virtual]

6.573.3 Member Function Documentation

6.573.3.1 virtual **commands::DataStructure*** **activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller::createObject** () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1331).

6.573.3.2 virtual unsigned char **activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller::getDataStructureType** () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

6.573

activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller

Class Reference

2573

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1336).

6.573.3.3 virtual void **activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller::looseMarshal** (**OpenWireFormat** * *wireFormat*, **commands::DataStructure** * *dataStructure*, **decaf::io::DataOutputStream** * *dataOut*) throw (**decaf::io::IOException**) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - **BinaryWriter** that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 631).

6.573.3.4 virtual void **activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller::looseUnmarshal** (**OpenWireFormat** * *wireFormat*, **commands::DataStructure** * *dataStructure*, **decaf::io::DataInputStream** * *dataIn*) throw (**decaf::io::IOException**) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - **BinaryReader** that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 632).

6.573.3.5 virtual int **activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller::tightMarshal** (**OpenWireFormat** * *wireFormat*, **commands::DataStructure** * *dataStructure*, **utils::BooleanStream** * *bs*) throw (**decaf::io::IOException**) [virtual]

Write the booleans that this object uses to a **BooleanStream**.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 633).

```
6.573.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller::tightMars
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 634).

```
6.573.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller::tightUnm
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.

6.574

activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller

Class Reference

2575

bs - `BooleanStream` stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 635).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/RemoveSubscriptionInfoMarshaller.h`

6.574 **activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller** Class Reference

Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 2572).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/RemoveSubscriptionInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller**:

Public Member Functions

- **RemoveSubscriptionInfoMarshaller** ()
- virtual **~RemoveSubscriptionInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

- virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.574.1 Detailed Description

Marshaling code for Open Wire Format for RemoveSubscriptionInfoMarshaller (p. 2572).
NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.574.2 Constructor & Destructor Documentation

6.574.2.1 activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller::RemoveSubscriptionInfoMarshaller () [inline]

6.574.2.2 virtual
activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller::~~RemoveSubscriptionInfoMarshaller () [inline, virtual]

6.574.3 Member Function Documentation

6.574.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements activemq::wireformat::openwire::marshal::DataStreamMarshaller (p. 1331).

6.574.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements activemq::wireformat::openwire::marshal::DataStreamMarshaller (p. 1336).

6.574

activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller

Class Reference

2577

```
6.574.3.3 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller::looseMars
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 611).

```
6.574.3.4 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller::looseUnm
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 612).

```
6.574.3.5 virtual int ac-
tivemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller::tightMars
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, utils::BooleanStream * bs ) throw (
decaf::io::IOException ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 613).

```
6.574.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller::tightMars
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 614).

```
6.574.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller::tightUnma
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 615).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/RemoveSubscriptionInfoMarshaller.h`

6.575 activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 2576).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/RemoveSubscriptionInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller**:

Public Member Functions

- **RemoveSubscriptionInfoMarshaller** ()
- virtual **~RemoveSubscriptionInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshall an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.575.1 Detailed Description

Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 2576).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.575.2 Constructor & Destructor Documentation

6.575.2.1 activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller::RemoveSubscriptionInfoMarshaller () [inline]

6.575.2.2 virtual
 activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller::~RemoveSubscriptionInfoMarshaller () [inline, virtual]

6.575.3 Member Function Documentation

6.575.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1331).

6.575.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1336).

6.575

activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller

Class Reference

2581

```
6.575.3.3 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller::looseMars
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 604).

```
6.575.3.4 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller::looseUnm
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 605).

```
6.575.3.5 virtual int ac-
tivemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller::tightMars
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, utils::BooleanStream * bs ) throw (
decaf::io::IOException ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 606).

```
6.575.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller::tightMars
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 608).

```
6.575.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller::tightUnma
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 609).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/RemoveSubscriptionInfoMarshaller.h`

6.576 activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 2580).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/RemoveSubscriptionInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller**:

Public Member Functions

- **RemoveSubscriptionInfoMarshaller** ()
- virtual **~RemoveSubscriptionInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshall an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.576.1 Detailed Description

Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 2580).
 NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.576.2 Constructor & Destructor Documentation

6.576.2.1 activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller::RemoveSubscriptionInfoMarshaller () [inline]

6.576.2.2 virtual
 activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller::~RemoveSubscriptionInfoMarshaller () [inline, virtual]

6.576.3 Member Function Documentation

6.576.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1331).

6.576.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1336).

6.576

activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller

Class Reference

2585

```
6.576.3.3 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller::looseMars
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 618).

```
6.576.3.4 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller::looseUnm
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 619).

```
6.576.3.5 virtual int ac-
tivemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller::tightMars
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, utils::BooleanStream * bs ) throw (
decaf::io::IOException ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 620).

```
6.576.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller::tightMars
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 621).

```
6.576.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller::tightUnma
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 622).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/RemoveSubscriptionInfoMarshaller.h`

6.577 activemq::commands::ReplayCommand Class Reference

```
#include <src/main/activemq/commands/ReplayCommand.h>
```

Inheritance diagram for `activemq::commands::ReplayCommand`:

Public Member Functions

- **ReplayCommand** ()
- virtual **~ReplayCommand** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaller share.
- virtual **ReplayCommand * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1372) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent.*
- virtual int **getFirstNakNumber** () const
- virtual void **setFirstNakNumber** (int firstNakNumber)
- virtual int **getLastNakNumber** () const
- virtual void **setLastNakNumber** (int lastNakNumber)
- virtual **Pointer< Command > visit** (activemq::state::CommandVisitor *visitor)
throw (exceptions::ActiveMQException)
Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_REPLAYCOMMAND** = 65

Protected Member Functions

- **ReplayCommand** (const **ReplayCommand** &)
- **ReplayCommand** & **operator=** (const **ReplayCommand** &)

Protected Attributes

- int **firstNakNumber**
- int **lastNakNumber**

6.577.1 Constructor & Destructor Documentation

6.577.1.1 **activemq::commands::ReplayCommand::ReplayCommand** (const **ReplayCommand** &) [inline, protected]

6.577.1.2 **activemq::commands::ReplayCommand::ReplayCommand** ()

6.577.1.3 **virtual** **activemq::commands::ReplayCommand::~~ReplayCommand** () [virtual]

6.577.2 Member Function Documentation

6.577.2.1 **virtual** **ReplayCommand*** **activemq::commands::ReplayCommand::cloneDataStructure** () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1373).

6.577.2.2 **virtual** void **activemq::commands::ReplayCommand::copyDataStructure** (const **DataStructure** * *src*) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 598).

6.577.2.3 virtual bool activemq::commands::ReplayCommand::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 598).

6.577.2.4 virtual unsigned char activemq::commands::ReplayCommand::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataStructure** (p. 1372) type copy.

Implements **activemq::commands::DataStructure** (p. 1375).

6.577.2.5 virtual int activemq::commands::ReplayCommand::getFirstNakNumber () const [virtual]**6.577.2.6 virtual int activemq::commands::ReplayCommand::getLastNakNumber () const** [virtual]**6.577.2.7 ReplayCommand& activemq::commands::ReplayCommand::operator= (const ReplayCommand &)** [inline, protected]**6.577.2.8 virtual void activemq::commands::ReplayCommand::setFirstNakNumber (int *firstNakNumber*)** [virtual]**6.577.2.9 virtual void activemq::commands::ReplayCommand::setLastNakNumber (int *lastNakNumber*)** [virtual]**6.577.2.10 virtual std::string activemq::commands::ReplayCommand::toString () const** [virtual]

Returns a string containing the information for this **DataStructure** (p. 1372) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 602).

```

6.577.2.11  virtual Pointer<Command> activemq::commands::ReplayCommand::visit (
               activemq::state::CommandVisitor * visitor ) throw (
               exceptions::ActiveMQException ) [virtual]

```

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

Returns

a **Response** (p. 2611) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 995).

6.577.3 Field Documentation

```

6.577.3.1  int activemq::commands::ReplayCommand::firstNakNumber [protected]

```

```

6.577.3.2  const unsigned char activemq::commands::ReplayCommand::ID _ -
             REPLAYCOMMAND = 65 [static]

```

```

6.577.3.3  int activemq::commands::ReplayCommand::lastNakNumber [protected]

```

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ReplayCommand.h`

6.578 activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 2587).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ReplayCommandMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller`:

Public Member Functions

- **ReplayCommandMarshaller** ()
- virtual **~ReplayCommandMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.

- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.578.1 Detailed Description

Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 2587). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.578.2 Constructor & Destructor Documentation

- 6.578.2.1 **activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller::ReplayCommandMarshaller** () [inline]
- 6.578.2.2 **virtual activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller::~~ReplayCommandMarshaller** () [inline, virtual]

6.578.3 Member Function Documentation

- 6.578.3.1 **virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller::createObject** () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.578.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller::getDataStructureType() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.578.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 631).

6.578.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 632).

6.578.3.5 `virtual int ac-`
`tivemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller::tightMarshal`
(`OpenWireFormat * wireFormat`, `commands::DataStructure`
`* dataStructure`, `utils::BooleanStream * bs`) throw (`decaf::io::IOException`) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller`
(p. 633).

6.578.3.6 `virtual void ac-`
`tivemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller::tightMarshal2`
(`OpenWireFormat * wireFormat`, `commands::DataStructure`
`* dataStructure`, `decaf::io::DataOutputStream * dataOut`,
`utils::BooleanStream * bs`) throw (`decaf::io::IOException`) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller`
(p. 634).

6.578.3.7 `virtual void ac-`
`tivemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller::tightUnmarshal`
(`OpenWireFormat * wireFormat`, `commands::DataStructure`
`* dataStructure`, `decaf::io::DataInputStream * dataIn`,
`utils::BooleanStream * bs`) throw (`decaf::io::IOException`) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 635).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ReplayCommandMarshaller.h`

6.579 `activemq::wireformat::openwire::marshal::v3::ReplayCommandMa` Class Reference

Marshaling code for Open Wire Format for `ReplayCommandMarshaller` (p. 2591).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ReplayCommandMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller`:

Public Member Functions

- `ReplayCommandMarshaller ()`
- `virtual ~ReplayCommandMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaler.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

Un-marshall an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.579.1 Detailed Description

Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p.2591). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.579.2 Constructor & Destructor Documentation

6.579.2.1 **activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller::ReplayCommandMarshaller** () [inline]

6.579.2.2 **virtual**
activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller::~~ReplayCommandMarshaller () [inline, virtual]

6.579.3 Member Function Documentation

6.579.3.1 **virtual commands::DataStructure*** **activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller::createObject** () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1331).

6.579.3.2 **virtual unsigned char** **activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller::getDataStructureType** () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.579.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 604).

6.579.3.4 `virtual void activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 605).

6.579.3.5 `virtual int activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller::tightMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 606).

```
6.579.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 608).

```
6.579.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 609).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ReplayCommandMarshaller.h`

6.580 **activemq::wireformat::openwire::marshal::v5::ReplayCommandMa** Class Reference

Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 2595).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ReplayCommandMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller**:

Public Member Functions

- **ReplayCommandMarshaller** ()
- virtual **~ReplayCommandMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

- virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.580.1 Detailed Description

Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 2595). NOTE! This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.580.2 Constructor & Destructor Documentation

6.580.2.1 activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller::ReplayCommandMarshaller () [inline]

6.580.2.2 virtual
 activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller::~~ReplayCommandMarshaller () [inline, virtual]

6.580.3 Member Function Documentation

6.580.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1331).

6.580.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1336).

6.580.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 624).

6.580.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 625).

6.580.3.5 `virtual int activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 626).

```
6.580.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 628).

```
6.580.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 629).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ReplayCommandMarshaller.h`

6.581 activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 2599).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ReplayCommandMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller**:

Public Member Functions

- **ReplayCommandMarshaller** ()
- virtual **~ReplayCommandMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.581.1 Detailed Description

Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p.2599). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.581.2 Constructor & Destructor Documentation

6.581.2.1 activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller::ReplayCommandMarshaller () [inline]

6.581.2.2 virtual
activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller::~~ReplayCommandMarshaller () [inline, virtual]

6.581.3 Member Function Documentation

6.581.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1331).

6.581.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1336).

6.581.3.3 virtual void activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 618).

6.581.3.4 virtual void activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 619).

6.581.3.5 virtual int activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 620).

```
6.581.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 621).

```
6.581.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 622).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/ReplayCommandMarshaller.h`

6.582 activemq::wireformat::openwire::marshal::v1::ReplayCommandMa Class Reference

Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 2603).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ReplayCommandMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller`:

Public Member Functions

- **ReplayCommandMarshaller** ()
- virtual **~ReplayCommandMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.582.1 Detailed Description

Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p.2603). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.582.2 Constructor & Destructor Documentation

6.582.2.1 activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller::ReplayCommandMarshaller () [inline]

6.582.2.2 virtual
activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller::~~ReplayCommandMarshaller () [inline, virtual]

6.582.3 Member Function Documentation

6.582.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1331).

6.582.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1336).

6.582.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 611).

6.582.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 612).

6.582.3.5 `virtual int activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 613).

```
6.582.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 614).

```
6.582.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 615).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/ReplayCommandMarshaller.h`

6.583 activemq::cmsutil::CmsTemplate::ResolveProducerExecutor Class Reference

```
#include <src/main/activemq/cmsutil/CmsTemplate.h>
```

Inheritance diagram for `activemq::cmsutil::CmsTemplate::ResolveProducerExecutor`:

Public Member Functions

- **ResolveProducerExecutor** (**ProducerCallback** **action*, **CmsTemplate** **parent*, const std::string &*destinationName*)
- virtual **~ResolveProducerExecutor** ()
- virtual **cms::Destination** * **getDestination** (**cms::Session** **session*) throw (**cms::CMSEException**)

6.583.1 Constructor & Destructor Documentation

6.583.1.1 `activemq::cmsutil::CmsTemplate::ResolveProducerExecutor::ResolveProducerExecutor (ProducerCallback * action, CmsTemplate * parent, const std::string & destinationName)` [inline]

6.583.1.2 `virtual
activemq::cmsutil::CmsTemplate::ResolveProducerExecutor::~~ResolveProducerExecutor ()` [inline, virtual]

6.583.2 Member Function Documentation

6.583.2.1 `virtual cms::Destination* activemq::cmsutil::CmsTemplate::ResolveProducerExecutor::getDestination (cms::Session * session) throw (cms::CMSEException)` [virtual]

The documentation for this class was generated from the following file:

- `src/main/activemq/cmsutil/CmsTemplate.h`

6.584 activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor Class Reference

```
#include <src/main/activemq/cmsutil/CmsTemplate.h>
```

Inheritance diagram for activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor:

Public Member Functions

- **ResolveReceiveExecutor** (**CmsTemplate** **parent*, const std::string &*selector*, bool *noLocal*, const std::string &*destinationName*)
- virtual **~ResolveReceiveExecutor** ()
- virtual **cms::Destination** * *getDestination* (**cms::Session** **session*) throw (**cms::CMSException**)

6.584.1 Constructor & Destructor Documentation

6.584.1.1 **activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor::ResolveReceiveExecutor** (**CmsTemplate** * *parent*, const std::string & *selector*, bool *noLocal*, const std::string & *destinationName*) [inline]

6.584.1.2 virtual **activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor::~~ResolveReceiveExecutor** () [inline, virtual]

6.584.2 Member Function Documentation

6.584.2.1 virtual **cms::Destination*** **activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor::getDestination** (**cms::Session** * *session*) throw (**cms::CMSException**) [virtual]

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/**CmsTemplate.h**

6.585 activemq::cmsutil::ResourceLifecycleManager Class Reference

Manages the lifecycle of a set of CMS resources.

```
#include <src/main/activemq/cmsutil/ResourceLifecycleManager.h>
```

Public Member Functions

- **ResourceLifecycleManager** ()
- virtual **~ResourceLifecycleManager** ()
Destructor - calls destroy

- void **addConnection** (**cms::Connection** *connection)
Adds a connection so that its life will be managed by this object.
- void **addSession** (**cms::Session** *session)
Adds a session so that its life will be managed by this object.
- void **addDestination** (**cms::Destination** *dest)
Adds a destination so that its life will be managed by this object.
- void **addMessageProducer** (**cms::MessageProducer** *producer)
Adds a message producer so that its life will be managed by this object.
- void **addMessageConsumer** (**cms::MessageConsumer** *consumer)
Adds a message consumer so that its life will be managed by this object.
- void **destroy** () throw (cms::CMSException)
Closes and destroys the contained CMS resources.
- void **releaseAll** ()
Releases all of the contained resources so that this object will no longer control their lifetimes.

6.585.1 Detailed Description

Manages the lifecycle of a set of CMS resources. A call to **destroy** will close and destroy all of the contained resources in the appropriate manner.

6.585.2 Constructor & Destructor Documentation

6.585.2.1 **activemq::cmsutil::ResourceLifecycleManager::ResourceLifecycleManager**
()

6.585.2.2 **virtual**
activemq::cmsutil::ResourceLifecycleManager::~~ResourceLifecycleManager
() [virtual]

Destructor - calls **destroy**

6.585.3 Member Function Documentation

6.585.3.1 **void activemq::cmsutil::ResourceLifecycleManager::addConnection** (
cms::Connection * *connection*) [inline]

Adds a connection so that its life will be managed by this object.

Parameters

connection the object to be managed

6.585.3.2 void activemq::cmsutil::ResourceLifecycleManager::addDestination (cms::Destination * *dest*) [inline]

Adds a destination so that its life will be managed by this object.

Parameters

dest the object to be managed

6.585.3.3 void activemq::cmsutil::ResourceLifecycleManager::addMessageConsumer (cms::MessageConsumer * *consumer*) [inline]

Adds a message consumer so that its life will be managed by this object.

Parameters

consumer the object to be managed

6.585.3.4 void activemq::cmsutil::ResourceLifecycleManager::addMessageProducer (cms::MessageProducer * *producer*) [inline]

Adds a message producer so that its life will be managed by this object.

Parameters

producer the object to be managed

6.585.3.5 void activemq::cmsutil::ResourceLifecycleManager::addSession (cms::Session * *session*) [inline]

Adds a session so that its life will be managed by this object.

Parameters

session the object to be managed

6.585.3.6 void activemq::cmsutil::ResourceLifecycleManager::destroy () throw (cms::CMSEException)

Closes and destroys the contained CMS resources.

Exceptions

cms::CMSEException (p. 960) thrown if an error occurs.

6.585.3.7 void activemq::cmsutil::ResourceLifecycleManager::releaseAll ()

Releases all of the contained resources so that this object will no longer control their lifetimes.

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/**ResourceLifecycleManager.h**

6.586 activemq::commands::Response Class Reference

```
#include <src/main/activemq/commands/Response.h>
```

Inheritance diagram for activemq::commands::Response:

Public Member Functions

- **Response** ()
- virtual **~Response** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **Response * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1372) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent.*
- virtual int **getCorrelationId** () const
- virtual void **setCorrelationId** (int correlationId)
- virtual bool **isResponse** () const
- virtual **Pointer< Command > visit** (activemq::state::CommandVisitor *visitor) throw (exceptions::ActiveMQException)
Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_RESPONSE** = 30

Protected Member Functions

- **Response** (const **Response** &)
- **Response** & **operator=** (const **Response** &)

Protected Attributes

- int **correlationId**

6.586.1 Constructor & Destructor Documentation

6.586.1.1 **activemq::commands::Response::Response** (const **Response** &)
[inline, protected]

6.586.1.2 **activemq::commands::Response::Response** ()

6.586.1.3 **virtual activemq::commands::Response::~~Response** () [virtual]

6.586.2 Member Function Documentation

6.586.2.1 **virtual Response*** **activemq::commands::Response::cloneDataStructure** () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1373).

Reimplemented in **activemq::commands::DataArrayResponse** (p. 1269), **activemq::commands::DataResponse** (p. 1308), **activemq::commands::ExceptionResponse** (p. 1485), and **activemq::commands::IntegerResponse** (p. 1668).

6.586.2.2 **virtual void activemq::commands::Response::copyDataStructure** (const **DataStructure** * *src*) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 598).

Reimplemented in **activemq::commands::DataArrayResponse** (p. 1269), **activemq::commands::DataResponse** (p. 1308), **activemq::commands::ExceptionResponse** (p. 1485), and **activemq::commands::IntegerResponse** (p. 1668).

6.586.2.3 `virtual bool activemq::commands::Response::equals (const
DataStructure * value) const` [virtual]

Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 598).

Reimplemented in **activemq::commands::DataArrayResponse** (p. 1269), **activemq::commands::DataResponse** (p. 1309), **activemq::commands::ExceptionResponse** (p. 1485), and **activemq::commands::IntegerResponse** (p. 1668).

6.586.2.4 `virtual int activemq::commands::Response::getCorrelationId () const` [virtual]

6.586.2.5 `virtual unsigned char activemq::commands::Response::getDataStructureType () const` [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataStructure** (p. 1372) type copy.

Implements **activemq::commands::DataStructure** (p. 1375).

Reimplemented in **activemq::commands::DataArrayResponse** (p. 1270), **activemq::commands::DataResponse** (p. 1309), **activemq::commands::ExceptionResponse** (p. 1486), and **activemq::commands::IntegerResponse** (p. 1669).

6.586.2.6 `virtual bool activemq::commands::Response::isResponse () const` [inline, virtual]

Returns

an answer of true to the **isResponse()** (p. 2613) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 601).

6.586.2.7 `Response& activemq::commands::Response::operator= (const Response &)` [inline, protected]

6.586.2.8 `virtual void activemq::commands::Response::setCorrelationId (int correlationId)` [virtual]

6.586.2.9 `virtual std::string activemq::commands::Response::toString () const` [virtual]

Returns a string containing the information for this **DataSet** (p.1372) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 602).

Reimplemented in **activemq::commands::DataArrayResponse** (p. 1270), **activemq::commands::DataResponse** (p. 1309), **activemq::commands::ExceptionResponse** (p. 1486), and **activemq::commands::IntegerResponse** (p. 1669).

6.586.2.10 `virtual Pointer<Command> activemq::commands::Response::visit (activemq::state::CommandVisitor * visitor) throw (exceptions::ActiveMQException)` [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

Returns

a **Response** (p. 2611) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 995).

6.586.3 Field Documentation

6.586.3.1 `int activemq::commands::Response::correlationId` [protected]

6.586.3.2 `const unsigned char activemq::commands::Response::ID_RESPONSE = 30` [static]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/Response.h`

6.587 activemq::transport::mock::ResponseBuilder Class Reference

Interface for all Protocols to implement that defines the behavior of the Broker in response to messages of that protocol.

```
#include <src/main/activemq/transport/mock/ResponseBuilder.h>
```

Inheritance diagram for `activemq::transport::mock::ResponseBuilder`:

Public Member Functions

- virtual `~ResponseBuilder()`
- virtual `Pointer< Response > buildResponse (const Pointer< Command > &command)=0`
Given a Command, check if it requires a response and return the appropriate Response that the Broker would send for this Command.
- virtual void `buildIncomingCommands (const Pointer< Command > &command, decaf::util::StlQueue< Pointer< Command > > &queue)=0`
*When called the **ResponseBuilder** (p. 2614) must construct all the Responses or Asynchronous commands that would be sent to this client by the Broker upon receipt of the passed command.*

6.587.1 Detailed Description

Interface for all Protocols to implement that defines the behavior of the Broker in response to messages of that protocol.

6.587.2 Constructor & Destructor Documentation

6.587.2.1 virtual `activemq::transport::mock::ResponseBuilder::~ResponseBuilder ()` [inline, virtual]

6.587.3 Member Function Documentation

6.587.3.1 virtual void `activemq::transport::mock::ResponseBuilder::buildIncomingCommands (const Pointer< Command > & command, decaf::util::StlQueue< Pointer< Command > > & queue)` [pure virtual]

When called the **ResponseBuilder** (p. 2614) must construct all the Responses or Asynchronous commands that would be sent to this client by the Broker upon receipt of the passed command.

Parameters

- command* - The Command being sent to the Broker.
- queue* - Queue of Command sent back from the broker.

Implemented in `activemq::wireformat::openwire::OpenWireResponseBuilder` (p. 2314).

6.587.3.2 virtual **Pointer<Response>** **activemq::transport::mock::ResponseBuilder::buildResponse**
(**const Pointer< Command > & command**) [pure virtual]

Given a **Command**, check if it requires a response and return the appropriate **Response** that the **Broker** would send for this **Command**.

Parameters

command - The command to build a response for

Returns

A **Response** object pointer, or **NULL** if no response.

Implemented in **activemq::wireformat::openwire::OpenWireResponseBuilder** (p. 2314).

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/mock/ResponseBuilder.h`

6.588 activemq::transport::correlator::ResponseCorrelator Class Reference

This type of transport filter is responsible for correlating asynchronous responses with requests.

`#include <src/main/activemq/transport/correlator/ResponseCorrelator.h>`

Inheritance diagram for **activemq::transport::correlator::ResponseCorrelator**:

Public Member Functions

- **ResponseCorrelator** (**const Pointer< Transport > &next**)
Constructor.
- virtual **~ResponseCorrelator** ()
- virtual void **oneway** (**const Pointer< Command > &command**) throw (**decaf::io::IOException**, **decaf::lang::exceptions::UnsupportedOperationException**)
Sends a one-way command.
- virtual **Pointer< Response > request** (**const Pointer< Command > &command**) throw (**decaf::io::IOException**, **decaf::lang::exceptions::UnsupportedOperationException**)
Sends the given request to the server and waits for the response.
- virtual **Pointer< Response > request** (**const Pointer< Command > &command**, **unsigned int timeout**) throw (**decaf::io::IOException**, **decaf::lang::exceptions::UnsupportedOperationException**)
Sends the given request to the server and waits for the response.
- virtual void **onCommand** (**const Pointer< Command > &command**)

This is called in the context of the nested transport's reading thread.

- virtual void **start** () throw (decaf::io::IOException)
Starts this transport object and creates the thread for polling on the input stream for commands.
- virtual void **close** () throw (decaf::io::IOException)
Stops the polling thread and closes the streams.
- virtual void **onTransportException** (Transport *source, const decaf::lang::Exception &ex)
Event handler for an exception from a command transport.

6.588.1 Detailed Description

This type of transport filter is responsible for correlating asynchronous responses with requests. Non-response messages are simply sent directly to the CommandListener. It owns the transport that it

6.588.2 Constructor & Destructor Documentation

6.588.2.1 activemq::transport::correlator::ResponseCorrelator::ResponseCorrelator (const Pointer< Transport > & next)

Constructor.

Parameters

next the next transport in the chain

6.588.2.2 virtual activemq::transport::correlator::ResponseCorrelator::~ResponseCorrelator () [virtual]

6.588.3 Member Function Documentation

6.588.3.1 virtual void activemq::transport::correlator::ResponseCorrelator::close () throw (decaf::io::IOException) [virtual]

Stops the polling thread and closes the streams.

This can be called explicitly, but is also called in the destructor. Once this object has been closed, it cannot be restarted.

Exceptions

IOException if errors occur.

Reimplemented from **activemq::transport::TransportFilter** (p. 3075).

6.588.3.2 virtual void activemq::transport::correlator::ResponseCorrelator::onCommand (const Pointer< Command > & *command*) [virtual]

This is called in the context of the nested transport's reading thread.

In the case of a response object, updates the request map and notifies those waiting on the response. Non-response messages are just delegated to the command listener.

Parameters

command the received from the nested transport.

Reimplemented from **activemq::transport::TransportFilter** (p. 3077).

6.588.3.3 virtual void activemq::transport::correlator::ResponseCorrelator::oneway (const Pointer< Command > & *command*) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]

Sends a one-way command.

Does not wait for any response from the broker.

Parameters

command the command to be sent.

Exceptions

IOException if an exception occurs during writing of the command.

UnsupportedOperationException if this method is not implemented by this transport.

Reimplemented from **activemq::transport::TransportFilter** (p. 3077).

6.588.3.4 virtual void activemq::transport::correlator::ResponseCorrelator::onTransportException (Transport * *source*, const decaf::lang::Exception & *ex*) [virtual]

Event handler for an exception from a command transport.

Parameters

source The source of the exception

ex The exception.

6.588.3.5 virtual Pointer<Response> activemq::transport::correlator::ResponseCorrelator::request (const Pointer< Command > & *command*) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]

Sends the given request to the server and waits for the response.

Parameters

command The request to send.

Returns

the response from the server.

Exceptions

IOException if an error occurs with the request.

Reimplemented from **activemq::transport::TransportFilter** (p. 3079).

```
6.588.3.6 virtual Pointer<Response> ac-
    tivemq::transport::correlator::ResponseCorrelator::request
    ( const Pointer< Command > & command, un-
      signed int timeout ) throw ( decaf::io::IOException,
      decaf::lang::exceptions::UnsupportedOperationException ) [virtual]
```

Sends the given request to the server and waits for the response.

Parameters

command The request to send.

timeout The time to wait for a response.

Returns

the response from the server.

Exceptions

IOException if an error occurs with the request.

Reimplemented from **activemq::transport::TransportFilter** (p. 3078).

```
6.588.3.7 virtual void activemq::transport::correlator::ResponseCorrelator::start (
    ) throw ( decaf::io::IOException ) [virtual]
```

Starts this transport object and creates the thread for polling on the input stream for commands.

If this object has been closed, throws an exception. Before calling start, the caller must set the IO streams and the reader and writer objects.

Exceptions

IOException if an error occurs or if this transport has already been closed.

Reimplemented from **activemq::transport::TransportFilter** (p. 3079).

The documentation for this class was generated from the following file:

- src/main/activemq/transport/correlator/**ResponseCorrelator.h**

6.589 activemq::wireformat::openwire::marshal::v2::ResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 2620).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ResponseMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ResponseMarshaller:

Public Member Functions

- **ResponseMarshaller** ()
- virtual **~ResponseMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.589.1 Detailed Description

Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 2620). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.589.2 Constructor & Destructor Documentation

6.589.2.1 `activemq::wireformat::openwire::marshal::v2::ResponseMarshaller::ResponseMarshaller () [inline]`

6.589.2.2 `virtual
activemq::wireformat::openwire::marshal::v2::ResponseMarshaller::~~ResponseMarshaller () [inline, virtual]`

6.589.3 Member Function Documentation

6.589.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ResponseMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller` (p. 1288), `activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller` (p. 1327), `activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller` (p. 1488), and `activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller` (p. 1671).

6.589.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ResponseMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller` (p. 1288), `activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller` (p. 1327), `activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller` (p. 1488), and `activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller` (p. 1671).

6.589.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ResponseMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 631).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller` (p. 1288), `activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller` (p. 1328), `activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller` (p. 1488), and `activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller` (p. 1671).

```
6.589.3.4 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::ResponseMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 632).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller` (p. 1289), `activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller` (p. 1328), `activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller` (p. 1489), and `activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller` (p. 1672).

```
6.589.3.5 virtual int ac-
tivemq::wireformat::openwire::marshal::v2::ResponseMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, utils::BooleanStream * bs ) throw (
decaf::io::IOException ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 633).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller` (p. 1289), `activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller` (p. 1328), `activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller` (p. 1489), and `activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller` (p. 1672).

6.589.3.6 `virtual void activemq::wireformat::openwire::marshal::v2::ResponseMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 634).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller` (p. 1290), `activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller` (p. 1329), `activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller` (p. 1490), and `activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller` (p. 1673).

```
6.589.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::ResponseMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 635).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller** (p. 1290), **activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller** (p. 1329), **activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller** (p. 1490), and **activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller** (p. 1673).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ResponseMarshaller.h`

6.590 activemq::wireformat::openwire::marshal::v3::ResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 2624).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ResponseMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller**:

Public Member Functions

- **ResponseMarshaller** ()
- virtual **~ResponseMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.590.1 Detailed Description

Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 2624). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.590.2 Constructor & Destructor Documentation

6.590.2.1 activemq::wireformat::openwire::marshal::v3::ResponseMarshaller::ResponseMarshaller () [inline]

6.590.2.2 virtual activemq::wireformat::openwire::marshal::v3::ResponseMarshaller::~~ResponseMarshaller () [inline, virtual]

6.590.3 Member Function Documentation

6.590.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ResponseMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1331).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller** (p. 1272), **activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller** (p. 1311), **activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller** (p. 1496), and **activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller** (p. 1679).

6.590.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::ResponseMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1336).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller** (p. 1272), **activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller** (p. 1311), **activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller** (p. 1496), and **activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller** (p. 1679).

6.590.3.3 virtual void activemq::wireformat::openwire::marshal::v3::ResponseMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 604).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller** (p. 1272), **activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller** (p. 1312), **activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller** (p. 1496), and **activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller** (p. 1679).

```

6.590.3.4 virtual void activemq::wireformat::openwire::marshal::v3::ResponseMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]

```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 605).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller` (p. 1273), `activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller` (p. 1312), `activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller` (p. 1497), and `activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller` (p. 1680).

```

6.590.3.5 virtual int activemq::wireformat::openwire::marshal::v3::ResponseMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, utils::BooleanStream * bs ) throw (
decaf::io::IOException ) [virtual]

```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 606).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller` (p. 1273), `activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller` (p. 1312), `activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller` (p. 1497), and `activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller` (p. 1680).

```
6.590.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::ResponseMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 608).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller` (p. 1274), `activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller` (p. 1313), `activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller` (p. 1498), and `activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller` (p. 1681).

```
6.590.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::ResponseMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 609).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller` (p. 1274), `activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller` (p. 1313), `activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller`

(p. 1498), and `activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller` (p. 1681).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ResponseMarshaller.h`

6.591 `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller` Class Reference

Marshaling code for Open Wire Format for `ResponseMarshaller` (p. 2628).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ResponseMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller`:

Public Member Functions

- `ResponseMarshaller ()`
- `virtual ~ResponseMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaler.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.
- `virtual void tightMarshal2 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write a object instance to data output stream.
- `virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)`

Write a object instance to data output stream.

6.591.1 Detailed Description

Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 2628). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.591.2 Constructor & Destructor Documentation

6.591.2.1 `activemq::wireformat::openwire::marshal::v5::ResponseMarshaller::ResponseMarshaller () [inline]`

6.591.2.2 `virtual
activemq::wireformat::openwire::marshal::v5::ResponseMarshaller::~~ResponseMarshaller () [inline, virtual]`

6.591.3 Member Function Documentation

6.591.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ResponseMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller` (p. 1284), `activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller` (p. 1319), `activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller` (p. 1504), and `activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller` (p. 1687).

6.591.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ResponseMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller` (p. 1284), `activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller` (p. 1319), `activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller`

(p. 1504), and `activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller` (p. 1687).

6.591.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ResponseMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 624).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller` (p. 1284), `activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller` (p. 1320), `activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller` (p. 1504), and `activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller` (p. 1687).

6.591.3.4 `virtual void activemq::wireformat::openwire::marshal::v5::ResponseMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 625).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller` (p. 1285), `activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller` (p. 1320), `activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller` (p. 1505), and `activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller` (p. 1688).

```
6.591.3.5 virtual int ac-
tivemq::wireformat::openwire::marshal::v5::ResponseMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, utils::BooleanStream * bs ) throw (
decaf::io::IOException ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 626).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller` (p. 1285), `activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller` (p. 1320), `activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller` (p. 1505), and `activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller` (p. 1688).

```
6.591.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::ResponseMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 628).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller` (p. 1286), `activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller`

(p. 1321), `activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller` (p. 1506), and `activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller` (p. 1689).

6.591.3.7 `virtual void activemq::wireformat::openwire::marshal::v5::ResponseMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 629).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller` (p. 1286), `activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller` (p. 1321), `activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller` (p. 1506), and `activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller` (p. 1689).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ResponseMarshaller.h`

6.592 `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller` Class Reference

Marshaling code for Open Wire Format for `ResponseMarshaller` (p. 2633).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ResponseMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller`:

Public Member Functions

- `ResponseMarshaller ()`
- `virtual ~ResponseMarshaller ()`
- `virtual commands::DataStructure * createObject () const`

Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.592.1 Detailed Description

Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 2633). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.592.2 Constructor & Destructor Documentation

6.592.2.1 `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller::ResponseMarshaller () [inline]`

6.592.2.2 `virtual
activemq::wireformat::openwire::marshal::v1::ResponseMarshaller::~~ResponseMarshaller () [inline, virtual]`

6.592.3 Member Function Documentation

6.592.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ResponseMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller` (p. 1280), `activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller` (p. 1315), `activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller` (p. 1492), and `activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller` (p. 1675).

6.592.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ResponseMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller` (p. 1280), `activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller` (p. 1315), `activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller` (p. 1492), and `activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller` (p. 1675).

6.592.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ResponseMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 611).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller` (p. 1280), `activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller` (p. 1316), `activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller` (p. 1492), and `activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller` (p. 1675).

```
6.592.3.4 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::ResponseMarshaller::looseUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure *
dataStructure, decaf::io::DataInputStream * dataIn ) throw (
decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 612).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller` (p. 1281), `activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller` (p. 1316), `activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller` (p. 1493), and `activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller` (p. 1676).

```
6.592.3.5 virtual int ac-
tivemq::wireformat::openwire::marshal::v1::ResponseMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, utils::BooleanStream * bs ) throw (
decaf::io::IOException ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 613).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller` (p. 1281), `activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller` (p. 1316), `activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller` (p. 1493), and `activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller` (p. 1676).

6.592.3.6 virtual void `activemq::wireformat::openwire::marshal::v1::ResponseMarshaller::tightMarshal2`
 (`OpenWireFormat * wireFormat`, `commands::DataStructure`
 * `dataStructure`, `decaf::io::DataOutputStream * dataOut`,
`utils::BooleanStream * bs`) throw (`decaf::io::IOException`) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 614).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller` (p. 1282), `activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller` (p. 1317), `activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller` (p. 1494), and `activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller` (p. 1677).


```

6.592.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::ResponseMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]

```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 615).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller** (p. 1282), **activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller** (p. 1317), **activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller** (p. 1494), and **activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller** (p. 1677).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/ResponseMarshaller.h`

6.593 activemq::wireformat::openwire::marshal::v4::ResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 2637).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ResponseMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller**:

Public Member Functions

- **ResponseMarshaller** ()
- virtual **~ResponseMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.593.1 Detailed Description

Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 2637). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.593.2 Constructor & Destructor Documentation

6.593.2.1 activemq::wireformat::openwire::marshal::v4::ResponseMarshaller::ResponseMarshaller () [inline]

6.593.2.2 virtual
activemq::wireformat::openwire::marshal::v4::ResponseMarshaller::~~ResponseMarshaller () [inline, virtual]

6.593.3 Member Function Documentation

6.593.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ResponseMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1331).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller** (p. 1276), **activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller** (p. 1323), **activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller** (p. 1500), and **activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller** (p. 1683).

6.593.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v4::ResponseMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1336).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller** (p. 1276), **activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller** (p. 1323), **activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller** (p. 1500), and **activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller** (p. 1683).

6.593.3.3 virtual void activemq::wireformat::openwire::marshal::v4::ResponseMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 618).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller** (p. 1276), **activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller** (p. 1324), **activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller** (p. 1500), and **activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller** (p. 1683).

6.593.3.4 virtual void activemq::wireformat::openwire::marshal::v4::ResponseMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 619).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller` (p. 1277), `activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller` (p. 1324), `activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller` (p. 1501), and `activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller` (p. 1684).

6.593.3.5 virtual int activemq::wireformat::openwire::marshal::v4::ResponseMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 620).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller` (p. 1277), `activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller` (p. 1324), `activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller` (p. 1501), and `activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller` (p. 1684).

6.593.3.6 virtual void activemq::wireformat::openwire::marshal::v4::ResponseMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 621).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller** (p. 1278), **activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller** (p. 1325), **activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller** (p. 1502), and **activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller** (p. 1685).

6.593.3.7 virtual void activemq::wireformat::openwire::marshal::v4::ResponseMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 622).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller** (p. 1278), **activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller** (p. 1325), **activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller**

(p. 1502), and `activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller` (p. 1685).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/ResponseMarshaller.h`

6.594 decaf::lang::Runnable Class Reference

Interface for a runnable object - defines a task that can be run by a thread.

```
#include <src/main/decaf/lang/Runnable.h>
```

Inheritance diagram for `decaf::lang::Runnable`:

Public Member Functions

- virtual `~Runnable()`
- virtual void `run()` = 0

Run method - called by the Thread class in the context of the thread.

6.594.1 Detailed Description

Interface for a runnable object - defines a task that can be run by a thread.

6.594.2 Constructor & Destructor Documentation

6.594.2.1 virtual `decaf::lang::Runnable::~~Runnable()` [inline, virtual]

6.594.3 Member Function Documentation

6.594.3.1 virtual void `decaf::lang::Runnable::run()` [pure virtual]

Run method - called by the Thread class in the context of the thread.

Implemented in `activemq::threads::CompositeTaskRunner` (p. 1018), `activemq::threads::DedicatedTaskRunner` (p. 1383), `activemq::transport::inactivity::ReadChecker` (p. 2518), `activemq::transport::inactivity::WriteChecker` (p. 3186), and `activemq::transport::IOTransport` (p. 1714).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Runnable.h`

6.595 decaf::lang::Runtime Class Reference

```
#include <src/main/decaf/lang/Runtime.h>
```

Inheritance diagram for decaf::lang::Runtime:

Public Member Functions

- virtual `~Runtime ()`

Static Public Member Functions

- static `Runtime * getRuntime ()`
*Gets the single instance of the Decaf **Runtime** (p. 2643) for this Process.*
- static void `initializeRuntime (int argc, char **argv)`
Initialize the Decaf Library passing it the args that were passed to the application at startup.
- static void `initializeRuntime ()`
Initialize the Decaf Library.
- static void `shutdownRuntime ()`
Shutdown the Decaf Library, this call should take places after all objects that were created from the Decaf library have been deallocated.

6.595.1 Constructor & Destructor Documentation

6.595.1.1 virtual `decaf::lang::Runtime::~~Runtime ()` [inline, virtual]

6.595.2 Member Function Documentation

6.595.2.1 static `Runtime* decaf::lang::Runtime::getRuntime ()` [static]

Gets the single instance of the Decaf **Runtime** (p. 2643) for this Process.

Returns

pointer to the single Decaf **Runtime** (p. 2643) instance that exists for this process

6.595.2.2 static void `decaf::lang::Runtime::initializeRuntime (int argc, char **argv)` [static]

Initialize the Decaf Library passing it the args that were passed to the application at startup.

Parameters

argc - The number of args passed
argv - Array of char* values passed to the Process on start.

Exceptions

runtime_error if the library is already initialized or an error occurs during initialization.

6.595.2.3 static void decaf::lang::Runtime::initializeRuntime () [static]

Initialize the Decaf Library.

Exceptions

runtime_error if the library is already initialized or an error occurs during initialization.

6.595.2.4 static void decaf::lang::Runtime::shutdownRuntime () [static]

Shutdown the Decaf Library, this call should take places after all objects that were created from the Decaf library have been deallocated.

Exceptions

runtime_error if the library has not already been initialized or an error occurs during shutdown.

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**Runtime.h**

6.596 decaf::lang::exceptions::RuntimeException Class Reference

```
#include <src/main/decaf/lang/exceptions/RuntimeException.h>
```

Inheritance diagram for decaf::lang::exceptions::RuntimeException:

Public Member Functions

- **RuntimeException** () throw ()
Default Constructor.
- **RuntimeException** (const **Exception** &ex) throw ()
Conversion Constructor from some other ActiveMQException.
- **RuntimeException** (const **RuntimeException** &ex) throw ()
Copy Constructor.
- **RuntimeException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **RuntimeException** (const std::exception *cause) throw ()
Constructor.

- **RuntimeException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **RuntimeException** * clone () const
Clones this exception.
- virtual ~**RuntimeException** () throw ()

6.596.1 Constructor & Destructor Documentation

6.596.1.1 decaf::lang::exceptions::RuntimeException::RuntimeException () throw () [inline]

Default Constructor.

6.596.1.2 decaf::lang::exceptions::RuntimeException::RuntimeException (const Exception & ex) throw () [inline]

Conversion Constructor from some other ActiveMQException.

Parameters

ex The **Exception** (p. 1476) whose data is to be copied into this one.

6.596.1.3 decaf::lang::exceptions::RuntimeException::RuntimeException (const RuntimeException & ex) throw () [inline]

Copy Constructor.

Parameters

ex The **Exception** (p. 1476) whose data is to be copied into this one.

6.596.1.4 decaf::lang::exceptions::RuntimeException::RuntimeException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.596.1.5 `decaf::lang::exceptions::RuntimeException::RuntimeException (const std::exception * cause) throw () [inline]`

Constructor.

Parameters

***cause* Pointer** (p. 2343) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.596.1.6 `decaf::lang::exceptions::RuntimeException::RuntimeException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.596.1.7 `virtual decaf::lang::exceptions::RuntimeException::~~RuntimeException () throw () [inline, virtual]`

6.596.2 Member Function Documentation

6.596.2.1 `virtual RuntimeException* decaf::lang::exceptions::RuntimeException::clone () const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

an new **Exception** (p. 1476) that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1479).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/exceptions/RuntimeException.h`

6.597 decaf::security_provider::SecurityProvider Class Reference

```
#include <src/main/decaf/security_provider/SecurityProvider.h>
```

Public Member Functions

- virtual `~SecurityProvider()`
- virtual `X500Principal * createX500Principal (const std::string &name)=0`
- virtual `X500Principal * createX500Principal (InputStream &is)=0`
- virtual `X500Principal * createX500Principal (unsigned char *buffer, int offset, int len)=0`

6.597.1 Constructor & Destructor Documentation

- 6.597.1.1** virtual `decaf::security_provider::SecurityProvider::~~SecurityProvider()` [`inline`, `virtual`]

6.597.2 Member Function Documentation

- 6.597.2.1** virtual `X500Principal* decaf::security_provider::SecurityProvider::createX500Principal (const std::string & name)` [`pure virtual`]
- 6.597.2.2** virtual `X500Principal* decaf::security_provider::SecurityProvider::createX500Principal (unsigned char * buffer, int offset, int len)` [`pure virtual`]
- 6.597.2.3** virtual `X500Principal* decaf::security_provider::SecurityProvider::createX500Principal (InputStream & is)` [`pure virtual`]

The documentation for this class was generated from the following file:

- `src/main/decaf/security_provider/SecurityProvider.h`

6.598 decaf::security_provider::SecurityProviderMap Class Reference

Lookup Map for Connector Factories.

```
#include <src/main/decaf/security_provider/SecurityProviderMap.h>
```

Public Member Functions

- void **registerSecurityProvider** (const std::string &name, **SecurityProvider** *provider)
Registers a new provider with this map.
- void **unregisterSecurityProvider** (const std::string &name)
Unregisters a provider from this map.
- **SecurityProvider * lookup** (const std::string &name)
Lookup the named provider in the Map.
- std::size_t **getSecurityProviderNames** (std::vector< std::string > &providers)
Fetch a list of provider names that this Map contains.

Static Public Member Functions

- static **SecurityProviderMap** * **getInstance** ()

Gets a singleton instance of this class.

6.598.1 Detailed Description

Lookup Map for Connector Factories. Use the Connector name to find the associated factory. This class does not take ownership of the stored factories, they must be deallocated somewhere.

6.598.2 Member Function Documentation

- 6.598.2.1** static **SecurityProviderMap*** **decaf::security_provider::SecurityProviderMap::getInstance** ()
[static]

Gets a singleton instance of this class.

Referenced by `decaf::security_provider::SecurityProviderRegistrar::SecurityProviderRegistrar()`, and `decaf::security_provider::SecurityProviderRegistrar::~~SecurityProviderRegistrar()`.

- 6.598.2.2** **std::size_t** **decaf::security_provider::SecurityProviderMap::getSecurityProviderNames** (**std::vector**< **std::string** > & *providers*)

Fetch a list of provider names that this Map contains.

Parameters

providers A vector object to receive the list

Returns

count of providers.

- 6.598.2.3** **SecurityProvider*** **decaf::security_provider::SecurityProviderMap::lookup** (const **std::string** & *name*)

Lookup the named provider in the Map.

Parameters

name the provider name to lookup

Returns

the provider associated with the name, or NULL

6.598.2.4 void decaf::security_provider::SecurityProviderMap::registerSecurityProvider (const std::string & *name*, SecurityProvider * *provider*)

Registers a new provider with this map.

Parameters

name A name to associate the provider with
provider the provider object to store in the map.

6.598.2.5 void decaf::security_provider::SecurityProviderMap::unregisterSecurityProvider (const std::string & *name*)

Unregisters a provider from this map.

Parameters

name the name of the provider to remove

The documentation for this class was generated from the following file:

- src/main/decaf/security_provider/SecurityProviderMap.h

6.599 decaf::security_provider::SecurityProviderRegistrar Class Reference

Registers the passed in provider into the provider map, this class can manage the lifetime of the registered provider (default behaviour).

```
#include <src/main/decaf/security_provider/SecurityProviderRegistrar.h>
```

Public Member Functions

- **SecurityProviderRegistrar** (const std::string &name, SecurityProvider *provider, bool manageLifetime=true)
Creates a registrar and registers the provider with the provider map.
- virtual ~**SecurityProviderRegistrar** ()
*Unregisters the provider from the provider map and destroys it if *manageLifetime* is set.*
- virtual SecurityProvider * **getProvider** ()
get a reference to the factory that this class is holding

6.599.1 Detailed Description

Registers the passed in provider into the provider map, this class can manage the lifetime of the registered provider (default behaviour).

6.599.2 Constructor & Destructor Documentation

6.599.2.1 `decaf::security_ - provider::SecurityProviderRegistrar::SecurityProviderRegistrar (const std::string & name, SecurityProvider * provider, bool manageLifetime = true) [inline]`

Creates a registrar and registers the provider with the provider map.

Parameters

name name of the provider to register

provider the provider object

manageLifetime boolean indicating if this object manages the lifetime of the factory that is being registered.

References `decaf::security_provider::SecurityProviderMap::getInstance()`.

6.599.2.2 `virtual decaf::security_ - provider::SecurityProviderRegistrar::~~SecurityProviderRegistrar () [inline, virtual]`

Unregisters the provider from the provider map and destroys it if `manageLifetime` is set.

References `decaf::security_provider::SecurityProviderMap::getInstance()`.

6.599.3 Member Function Documentation

6.599.3.1 `virtual SecurityProvider* decaf::security_ - provider::SecurityProviderRegistrar::getProvider () [inline, virtual]`

get a reference to the factory that this class is holding

Returns

reference to a factory class

The documentation for this class was generated from the following file:

- `src/main/decaf/security_provider/SecurityProviderRegistrar.h`

6.600 decaf::util::concurrent::Semaphore Class Reference

A counting semaphore.

```
#include <src/main/decaf/util/concurrent/Semaphore.h>
```

Public Member Functions

- `Semaphore (int permits)`

Creates a **Semaphore** (p. 2651) with the given number of permits and nonfair fairness setting.

- **Semaphore** (int permits, bool fair)
Creates a **Semaphore** (p. 2651) with the given number of permits and the given fairness setting.
- virtual **~Semaphore** ()
- void **acquire** () throw (decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::RuntimeException)
Acquires a permit from this semaphore, blocking until one is available, or the thread is interrupted.
- void **acquireUninterruptibly** () throw (decaf::lang::exceptions::RuntimeException)
Acquires a permit from this semaphore, blocking until one is available.
- bool **tryAcquire** () throw (decaf::lang::exceptions::RuntimeException)
Acquires a permit from this semaphore, only if one is available at the time of invocation.
- bool **tryAcquire** (long long timeout, const **TimeUnit** &unit) throw (decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::RuntimeException)
Acquires a permit from this semaphore, if one becomes available within the given waiting time and the current thread has not been interrupted.
- void **release** () throw (decaf::lang::exceptions::RuntimeException)
Releases a permit, returning it to the semaphore.
- void **acquire** (int permits) throw (decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::RuntimeException)
Acquires the given number of permits from this semaphore, blocking until all are available, or the thread is interrupted.
- void **acquireUninterruptibly** (int permits) throw (decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::RuntimeException)
Acquires the given number of permits from this semaphore, blocking until all are available.
- bool **tryAcquire** (int permits) throw (decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::RuntimeException)
Acquires the given number of permits from this semaphore, only if all are available at the time of invocation.
- bool **tryAcquire** (int permits, long long timeout, const **TimeUnit** &unit) throw (decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::RuntimeException)
Acquires the given number of permits from this semaphore, if all become available within the given waiting time and the current thread has not been interrupted.
- void **release** (int permits) throw (decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::RuntimeException)
Releases the given number of permits, returning them to the semaphore.
- int **availablePermits** () const

Returns the current number of permits available in this semaphore.

- `int drainPermits () throw (decaf::lang::exceptions::RuntimeException)`

Acquires and returns all permits that are immediately available.

- `bool isFair () const`
- `std::string toString () const`

Returns a string identifying this semaphore, as well as its state.

6.600.1 Detailed Description

A counting semaphore. Conceptually, a semaphore maintains a set of permits. Each **acquire()** (p. 2654) blocks if necessary until a permit is available, and then takes it. Each **release()** (p. 2657) adds a permit, potentially releasing a blocking acquirer. However, no actual permit objects are used; the **Semaphore** (p. 2651) just keeps a count of the number available and acts accordingly.

Semaphores are often used to restrict the number of threads than can access some (physical or logical) resource.

```
class Pool { private:
static const int MAX_AVAILABLE = 100; Semaphore (p. 2651) available;
std::vector<std::string> items; std::vector<bool> used;
Mutex (p. 2239) lock;
public:
Pool() : available( MAX_AVAILABLE, true ) { used.resize( MAX_AVAILABLE ); items.resize(
MAX_AVAILABLE ); }
std::string getItem() throws InterruptedException { available.acquire(); return getNextAvail-
ableItem(); }
void putItem( std::string x ) { if( markAsUnused(x) ) { available.release(); } }
std::string getNextAvailableItem() {
synchronized( &lock ) (p. 3620) { for( int i = 0; i < MAX_AVAILABLE; ++i ) { if( !used[i] )
{ used[i] = true; return items[i]; } }
return std::string(); // not reached }
bool markAsUnused( const std::string& item ) { synchronized( &lock ) (p. 3620) { for( int i =
0; i < MAX_AVAILABLE; ++i ) { if( item == items[i] ) { if( used[i] ) { used[i] = false; return
true; } else return false; } } } return false; } };
```

Before obtaining an item each thread must acquire a permit from the semaphore, guaranteeing that an item is available for use. When the thread has finished with the item it is returned back to the pool and a permit is returned to the semaphore, allowing another thread to acquire that item. Note that no synchronization lock is held when **acquire()** (p. 2654) is called as that would prevent an item from being returned to the pool. The semaphore encapsulates the synchronization needed to restrict access to the pool, separately from any synchronization needed to maintain the consistency of the pool itself.

A semaphore initialized to one, and which is used such that it only has at most one permit available, can serve as a mutual exclusion lock. This is more commonly known as a binary semaphore, because it only has two states: one permit available, or zero permits available. When used in this

way, the binary semaphore has the property (unlike many **Lock** (p. 1903) implementations), that the "lock" can be released by a thread other than the owner (as semaphores have no notion of ownership). This can be useful in some specialized contexts, such as deadlock recovery.

The constructor for this class optionally accepts a fairness parameter. When set false, this class makes no guarantees about the order in which threads acquire permits. In particular, barging is permitted, that is, a thread invoking **acquire()** (p. 2654) can be allocated a permit ahead of a thread that has been waiting - logically the new thread places itself at the head of the queue of waiting threads. When fairness is set true, the semaphore guarantees that threads invoking any of the acquire methods are selected to obtain permits in the order in which their invocation of those methods was processed (first-in-first-out; FIFO). Note that FIFO ordering necessarily applies to specific internal points of execution within these methods. So, it is possible for one thread to invoke acquire before another, but reach the ordering point after the other, and similarly upon return from the method. Also note that the untimed tryAcquire methods do not honor the fairness setting, but will take any permits that are available.

Generally, semaphores used to control resource access should be initialized as fair, to ensure that no thread is starved out from accessing a resource. When using semaphores for other kinds of synchronization control, the throughput advantages of non-fair ordering often outweigh fairness considerations.

This class also provides convenience methods to acquire and release multiple permits at a time. Beware of the increased risk of indefinite postponement when these methods are used without fairness set true.

Since

1.0

6.600.2 Constructor & Destructor Documentation

6.600.2.1 decaf::util::concurrent::Semaphore::Semaphore (int *permits*)

Creates a **Semaphore** (p. 2651) with the given number of permits and nonfair fairness setting.

Parameters

permits the initial number of permits available. This value may be negative, in which case releases must occur before any acquires will be granted.

6.600.2.2 decaf::util::concurrent::Semaphore::Semaphore (int *permits*, bool *fair*)

Creates a **Semaphore** (p. 2651) with the given number of permits and the given fairness setting.

Parameters

permits the initial number of permits available. This value may be negative, in which case releases must occur before any acquires will be granted.

fair true if this semaphore will guarantee first-in first-out granting of permits under contention, else false

6.600.2.3 `virtual decaf::util::concurrent::Semaphore::~~Semaphore () [virtual]`

6.600.3 Member Function Documentation

6.600.3.1 `void decaf::util::concurrent::Semaphore::acquire ()
throw (decaf::lang::exceptions::InterruptedException,
decaf::lang::exceptions::RuntimeException)`

Acquires a permit from this semaphore, blocking until one is available, or the thread is interrupted.

Acquires a permit, if one is available and returns immediately, reducing the number of available permits by one.

If no permit is available then the current thread becomes disabled for thread scheduling purposes and lies dormant until one of two things happens:

* Some other thread invokes the **release()** (p.2657) method for this semaphore and the current thread is next to be assigned a permit; or * Some other thread interrupts the current thread.

If the current thread:

* has its interrupted status set on entry to this method; or * is interrupted while waiting for a permit,

then `InterruptedException` is thrown and the current thread's interrupted status is cleared.

Exceptions

InterruptedException - if the current thread is interrupted.

RuntimeException if an unexpected error occurs while acquiring the **Semaphore** (p. 2651).

6.600.3.2 `void decaf::util::concurrent::Semaphore::acquire (int permits
) throw (decaf::lang::exceptions::InterruptedException,
decaf::lang::exceptions::IllegalArgumentException,
decaf::lang::exceptions::RuntimeException)`

Acquires the given number of permits from this semaphore, blocking until all are available, or the thread is interrupted.

Acquires the given number of permits, if they are available, and returns immediately, reducing the number of available permits by the given amount.

If insufficient permits are available then the current thread becomes disabled for thread scheduling purposes and lies dormant until one of two things happens:

* Some other thread invokes one of the release methods for this semaphore, the current thread is next to be assigned permits and the number of available permits satisfies this request; or * Some other thread interrupts the current thread.

If the current thread:

* has its interrupted status set on entry to this method; or * is interrupted while waiting for a permit,

then `InterruptedException` is thrown and the current thread's interrupted status is cleared. Any permits that were to be assigned to this thread are instead assigned to other threads trying to acquire permits, as if permits had been made available by a call to **release()** (p. 2657).

Parameters

permits the number of permits to acquire.

Exceptions

InterruptedException if the current thread is interrupted.

IllegalArgumentException if the permits argument is negative.

RuntimeException if an unexpected error occurs while acquiring the **Semaphore** (p. 2651).

6.600.3.3 void decaf::util::concurrent::Semaphore::acquireUninterruptibly () throw (decaf::lang::exceptions::RuntimeException)

Acquires a permit from this semaphore, blocking until one is available.

Acquires a permit, if one is available and returns immediately, reducing the number of available permits by one.

If no permit is available then the current thread becomes disabled for thread scheduling purposes and lies dormant until some other thread invokes the **release()** (p. 2657) method for this semaphore and the current thread is next to be assigned a permit.

If the current thread is interrupted while waiting for a permit then it will continue to wait, but the time at which the thread is assigned a permit may change compared to the time it would have received the permit had no interruption occurred. When the thread does return from this method its interrupt status will be set.

Exceptions

RuntimeException if an unexpected error occurs while acquiring the **Semaphore** (p. 2651).

6.600.3.4 void decaf::util::concurrent::Semaphore::acquireUninterruptibly (int *permits*) throw (decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::RuntimeException)

Acquires the given number of permits from this semaphore, blocking until all are available.

Acquires the given number of permits, if they are available, and returns immediately, reducing the number of available permits by the given amount.

If insufficient permits are available then the current thread becomes disabled for thread scheduling purposes and lies dormant until some other thread invokes one of the release methods for this semaphore, the current thread is next to be assigned permits and the number of available permits satisfies this request.

If the current thread is interrupted while waiting for permits then it will continue to wait and its position in the queue is not affected. When the thread does return from this method its interrupt status will be set.

Parameters

permits the number of permits to acquire.

Exceptions

IllegalArgumentException if the permits argument is negative.

RuntimeException if an unexpected error occurs while acquiring the **Semaphore** (p. 2651).

6.600.3.5 `int decaf::util::concurrent::Semaphore::availablePermits () const`

Returns the current number of permits available in this semaphore.

This method is typically used for debugging and testing purposes.

Returns

the number of permits available in this semaphore

6.600.3.6 `int decaf::util::concurrent::Semaphore::drainPermits () throw (decaf::lang::exceptions::RuntimeException)`

Acquires and returns all permits that are immediately available.

Returns

the number of permits acquired

Exceptions

RuntimeException if an unexpected error occurs while draining the **Semaphore** (p. 2651).

6.600.3.7 `bool decaf::util::concurrent::Semaphore::isFair () const`**Returns**

true if this semaphore has fairness set true

6.600.3.8 `void decaf::util::concurrent::Semaphore::release () throw (decaf::lang::exceptions::RuntimeException)`

Releases a permit, returning it to the semaphore.

Releases a permit, increasing the number of available permits by one. If any threads are trying to acquire a permit, then one is selected and given the permit that was just released. That thread is (re)enabled for thread scheduling purposes.

There is no requirement that a thread that releases a permit must have acquired that permit by calling **acquire()** (p. 2654). Correct usage of a semaphore is established by programming convention in the application.

Exceptions

RuntimeException if an unexpected error occurs while releasing the **Semaphore** (p. 2651).

6.600.3.9 `void decaf::util::concurrent::Semaphore::release (int permits) throw (decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::RuntimeException)`

Releases the given number of permits, returning them to the semaphore.

Releases the given number of permits, increasing the number of available permits by that amount. If any threads are trying to acquire permits, then one is selected and given the permits that were just released. If the number of available permits satisfies that thread's request then that thread is (re)enabled for thread scheduling purposes; otherwise the thread will wait until sufficient permits are available. If there are still permits available after this thread's request has been satisfied, then those permits are assigned in turn to other threads trying to acquire permits.

Parameters

permits the number of permits to release

Exceptions

IllegalArgumentException if the permits argument is negative.

RuntimeException if an unexpected error occurs while releasing the **Semaphore** (p. 2651).

6.600.3.10 std::string decaf::util::concurrent::Semaphore::toString () const

Returns a string identifying this semaphore, as well as its state.

The state, in brackets, includes the String "Permits =" followed by the number of permits.

Returns

a string identifying this semaphore, as well as its state

6.600.3.11 bool decaf::util::concurrent::Semaphore::tryAcquire (int permits, long long timeout, const TimeUnit & unit) throw (decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::RuntimeException)

Acquires the given number of permits from this semaphore, if all become available within the given waiting time and the current thread has not been interrupted.

Acquires the given number of permits, if they are available and returns immediately, with the value true, reducing the number of available permits by the given amount.

If insufficient permits are available then the current thread becomes disabled for thread scheduling purposes and lies dormant until one of three things happens:

* Some other thread invokes one of the release methods for this semaphore, the current thread is next to be assigned permits and the number of available permits satisfies this request; or * Some other thread interrupts the current thread; or * The specified waiting time elapses.

If the permits are acquired then the value true is returned.

If the current thread:

* has its interrupted status set on entry to this method; or * is interrupted while waiting to acquire the permits,

then InterruptedException is thrown and the current thread's interrupted status is cleared. Any permits that were to be assigned to this thread, are instead assigned to other threads trying to acquire permits, as if the permits had been made available by a call to **release()** (p. 2657).

If the specified waiting time elapses then the value false is returned. If the time is less than or equal to zero, the method will not wait at all. Any permits that were to be assigned to this thread,

are instead assigned to other threads trying to acquire permits, as if the permits had been made available by a call to **release()** (p. 2657).

Parameters

permits the number of permits to acquire
timeout the maximum amount of time to wait to acquire the permits.
unit the units that the timeout param represents.

Returns

true if all permits were acquired and false if the waiting time elapsed before all permits were acquired

Exceptions

IllegalArgumentException if the permits argument is negative.
RuntimeException if an unexpected error occurs while acquiring the **Semaphore** (p. 2651).

6.600.3.12 `bool decaf::util::concurrent::Semaphore::tryAcquire
 (long long timeout, const TimeUnit & unit)
 throw (decaf::lang::exceptions::InterruptedException,
 decaf::lang::exceptions::RuntimeException)`

Acquires a permit from this semaphore, if one becomes available within the given waiting time and the current thread has not been interrupted.

Acquires a permit, if one is available and returns immediately, with the value true, reducing the number of available permits by one.

If no permit is available then the current thread becomes disabled for thread scheduling purposes and lies dormant until one of three things happens:

- * Some other thread invokes the **release()** (p. 2657) method for this semaphore and the current thread is next to be assigned a permit; or
- * Some other thread interrupts the current thread; or
- * The specified waiting time elapses.

If a permit is acquired then the value true is returned.

If the current thread:

- * has its interrupted status set on entry to this method; or
- * is interrupted while waiting to acquire a permit,

then **InterruptedException** is thrown and the current thread's interrupted status is cleared.

If the specified waiting time elapses then the value false is returned. If the time is less than or equal to zero, the method will not wait at all.

Parameters

timeout the maximum time to wait for a permit
unit the time unit of the timeout argument

Returns

true if a permit was acquired and false if the waiting time elapsed before a permit was acquired

Exceptions

InterruptedException if the current thread is interrupted.

RuntimeException if an unexpected error occurs while acquiring the **Semaphore** (p. 2651).

6.600.3.13 `bool decaf::util::concurrent::Semaphore::tryAcquire () throw (decaf::lang::exceptions::RuntimeException)`

Acquires a permit from this semaphore, only if one is available at the time of invocation.

Acquires a permit, if one is available and returns immediately, with the value true, reducing the number of available permits by one.

If no permit is available then this method will return immediately with the value false.

Even when this semaphore has been set to use a fair ordering policy, a call to **tryAcquire()** (p. 2659) will immediately acquire a permit if one is available, whether or not other threads are currently waiting. This "barging" behavior can be useful in certain circumstances, even though it breaks fairness. If you want to honor the fairness setting, then use **tryAcquire(0, TimeUnit.SECONDS)** (p. 3014) which is almost equivalent (it also detects interruption).

Returns

true if a permit was acquired and false otherwise

Exceptions

RuntimeException if an unexpected error occurs while acquiring the **Semaphore** (p. 2651).

6.600.3.14 `bool decaf::util::concurrent::Semaphore::tryAcquire (int permits) throw (decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::RuntimeException)`

Acquires the given number of permits from this semaphore, only if all are available at the time of invocation.

Acquires the given number of permits, if they are available, and returns immediately, with the value true, reducing the number of available permits by the given amount.

If insufficient permits are available then this method will return immediately with the value false and the number of available permits is unchanged.

Even when this semaphore has been set to use a fair ordering policy, a call to **tryAcquire** will immediately acquire a permit if one is available, whether or not other threads are currently waiting. This "barging" behavior can be useful in certain circumstances, even though it breaks fairness. If you want to honor the fairness setting, then use **tryAcquire(permits, 0, TimeUnit.SECONDS)** (p. 3014) which is almost equivalent (it also detects interruption).

Parameters

permits the number of permits to acquire

Returns

true if the permits were acquired and false otherwise.

Exceptions

IllegalArgumentException if the permits argument is negative.

RuntimeException if an unexpected error occurs while acquiring the **Semaphore** (p. 2651).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/Semaphore.h`

6.601 activemq::cmsutil::CmsTemplate::SendExecutor Class Reference

```
#include <src/main/activemq/cmsutil/CmsTemplate.h>
```

Inheritance diagram for `activemq::cmsutil::CmsTemplate::SendExecutor`:

Public Member Functions

- **SendExecutor** (**MessageCreator** *messageCreator, **CmsTemplate** *parent)
- virtual **~SendExecutor** ()
- virtual void **doInCms** (**cms::Session** *session, **cms::MessageProducer** *producer) throw (**cms::CMSEException**)

Execute an action given a session and producer.

6.601.1 Constructor & Destructor Documentation

6.601.1.1 `activemq::cmsutil::CmsTemplate::SendExecutor::SendExecutor (MessageCreator * messageCreator, CmsTemplate * parent) [inline]`

6.601.1.2 `virtual activemq::cmsutil::CmsTemplate::SendExecutor::~~SendExecutor () [inline, virtual]`

6.601.2 Member Function Documentation

6.601.2.1 `virtual void activemq::cmsutil::CmsTemplate::SendExecutor::doInCms (cms::Session * session, cms::MessageProducer * producer) throw (cms::CMSEException) [inline, virtual]`

Execute an action given a session and producer.

Parameters

session the CMS Session

producer the CMS Producer

Exceptions

cms::CMSEException (p. 960) if thrown by CMS API methods

Implements **activemq::cmsutil::ProducerCallback** (p. 2444).

The documentation for this class was generated from the following file:

- `src/main/activemq/cmsutil/CmsTemplate.h`

6.602 decaf::net::ServerSocket Class Reference

A server socket class (for testing purposes).

```
#include <src/main/decaf/net/ServerSocket.h>
```

Public Types

- `typedef apr_socket_t * SocketHandle`
- `typedef apr_sockaddr_t * SocketAddress`

Public Member Functions

- **ServerSocket** ()
Constructor.
- virtual **~ServerSocket** ()
Destructor.
- virtual void **bind** (const char *host, int port) throw (SocketException)
Bind and listen to given IP/dns and port.
- virtual void **bind** (const char *host, int port, int backlog) throw (SocketException)
Bind and listen to given IP/dns and port.
- virtual **Socket** * **accept** () throw (SocketException)
Blocks until a client connects to the bound socket.
- virtual void **close** () throw (lang::Exception)
Closes the server socket.
- virtual bool **isBound** () const

6.602.1 Detailed Description

A server socket class (for testing purposes).

6.602.2 Member Typedef Documentation

6.602.2.1 `typedef apr_socket_t* decaf::net::ServerSocket::SocketAddress`

6.602.2.2 `typedef apr_socket_t* decaf::net::ServerSocket::SocketHandle`

6.602.3 Constructor & Destructor Documentation

6.602.3.1 `decaf::net::ServerSocket::ServerSocket ()`

Constructor.

Creates a non-bound server socket.

6.602.3.2 `virtual decaf::net::ServerSocket::~ServerSocket () [virtual]`

Destructor.

Releases socket handle if `close()` (p. 2663) hasn't been called.

6.602.4 Member Function Documentation

6.602.4.1 `virtual Socket* decaf::net::ServerSocket::accept () throw (SocketException) [virtual]`

Blocks until a client connects to the bound socket.

Returns

new socket. Never returns NULL.

6.602.4.2 `virtual void decaf::net::ServerSocket::bind (const char * host, int port, int backlog) throw (SocketException) [virtual]`

Bind and listen to given IP/dns and port.

Parameters

host IP address or host name.

port TCP port between 1..65535

backlog Size of listen backlog.

6.602.4.3 `virtual void decaf::net::ServerSocket::bind (const char * host, int port) throw (SocketException) [virtual]`

Bind and listen to given IP/dns and port.

Parameters

host IP address or host name.

port TCP port between 1..65535

6.602.4.4 `virtual void decaf::net::ServerSocket::close () throw (lang::Exception) [virtual]`

Closes the server socket.

6.602.4.5 `virtual bool decaf::net::ServerSocket::isBound () const [virtual]`

Returns

true of the server socket is bound.

The documentation for this class was generated from the following file:

- `src/main/decaf/net/ServerSocket.h`

6.603 cms::Session Class Reference

A **Session** (p.2663) object is a single-threaded context for producing and consuming messages.

```
#include <src/main/cms/Session.h>
```

Inheritance diagram for cms::Session:

Public Types

- enum **AcknowledgeMode** {
AUTO_ACKNOWLEDGE, **DUPS_OK_ACKNOWLEDGE**, **CLIENT_-**
ACKNOWLEDGE, **SESSION_TRANSACTIONED**,
INDIVIDUAL_ACKNOWLEDGE }

Public Member Functions

- virtual **~Session** ()
- virtual void **close** ()=0 throw (CMSEException)
Closes this session as well as any active child consumers or producers.
- virtual void **commit** ()=0 throw (CMSEException)
Commits all messages done in this transaction and releases any locks currently held.
- virtual void **rollback** ()=0 throw (CMSEException)
Rolls back all messages done in this transaction and releases any locks currently held.
- virtual void **recover** ()=0 throw (CMSEException)
Stops message delivery in this session, and restarts message delivery with the oldest unacknowledged message.
- virtual **MessageConsumer** * **createConsumer** (const **Destination** *destination)=0 throw (CMSEException)

*Creates a **MessageConsumer** (p. 2080) for the specified destination.*

- virtual **MessageConsumer** * **createConsumer** (const **Destination** *destination, const std::string &selector)=0 throw (CMSEException)

*Creates a **MessageConsumer** (p. 2080) for the specified destination, using a message selector.*

- virtual **MessageConsumer** * **createConsumer** (const **Destination** *destination, const std::string &selector, bool noLocal)=0 throw (CMSEException)

*Creates a **MessageConsumer** (p. 2080) for the specified destination, using a message selector.*

- virtual **MessageConsumer** * **createDurableConsumer** (const **Topic** *destination, const std::string &name, const std::string &selector, bool noLocal=false)=0 throw (CMSEException)

*Creates a durable subscriber to the specified topic, using a **Message** (p. 2036) selector.*

- virtual **MessageProducer** * **createProducer** (const **Destination** *destination)=0 throw (CMSEException)

*Creates a **MessageProducer** (p. 2189) to send messages to the specified destination.*

- virtual **QueueBrowser** * **createBrowser** (const **cms::Queue** *queue)=0 throw (CMSEException)

*Creates a new **QueueBrowser** (p. 2511) to peek at Messages on the given **Queue** (p. 2510).*

- virtual **QueueBrowser** * **createBrowser** (const **cms::Queue** *queue, const std::string &selector)=0 throw (CMSEException)

*Creates a new **QueueBrowser** (p. 2511) to peek at Messages on the given **Queue** (p. 2510).*

- virtual **Queue** * **createQueue** (const std::string &queueName)=0 throw (CMSEException)

*Creates a queue identity given a **Queue** (p. 2510) name.*

- virtual **Topic** * **createTopic** (const std::string &topicName)=0 throw (CMSEException)

*Creates a topic identity given a **Queue** (p. 2510) name.*

- virtual **TemporaryQueue** * **createTemporaryQueue** ()=0 throw (CMSEException)

*Creates a **TemporaryQueue** (p. 2971) object.*

- virtual **TemporaryTopic** * **createTemporaryTopic** ()=0 throw (CMSEException)

*Creates a **TemporaryTopic** (p. 2973) object.*

- virtual **Message** * **createMessage** ()=0 throw (CMSEException)

*Creates a new **Message** (p. 2036).*

- virtual **BytesMessage** * **createBytesMessage** ()=0 throw (CMSEException)

*Creates a **BytesMessage** (p. 874).*

- virtual **BytesMessage** * **createBytesMessage** (const unsigned char *bytes, std::size_t bytesize)=0 throw (CMSEException)

*Creates a **BytesMessage** (p. 874) and sets the payload to the passed value.*

- virtual **StreamMessage** * **createStreamMessage** ()=0 throw (CMSEException)

*Creates a new **StreamMessage** (p. 2892).*

- virtual **TextMessage** * **createTextMessage** ()=0 throw (CMSEException)
*Creates a new **TextMessage** (p. 2974).*
- virtual **TextMessage** * **createTextMessage** (const std::string &text)=0 throw (CMSEException)
*Creates a new **TextMessage** (p. 2974) and set the text to the value given.*
- virtual **MapMessage** * **createMapMessage** ()=0 throw (CMSEException)
*Creates a new **MapMessage** (p. 1982).*
- virtual **AcknowledgeMode** **getAcknowledgeMode** () const =0 throw (CMSEException)
Returns the acknowledgment mode of the session.
- virtual bool **isTransacted** () const =0 throw (CMSEException)
*Gets if the Sessions is a Transacted **Session** (p. 2663).*
- virtual void **unsubscribe** (const std::string &name)=0 throw (CMSEException)
Unsubscribes a durable subscription that has been created by a client.

6.603.1 Detailed Description

A **Session** (p.2663) object is a single-threaded context for producing and consuming messages. A session serves several purposes:

- It is a factory for its message producers and consumers.
- It supplies provider-optimized message factories.
- It is a factory for **TemporaryTopics** and **TemporaryQueues**.
- It provides a way to create **Queue** (p. 2510) or **Topic** (p. 3014) objects for those clients that need to dynamically manipulate provider-specific destination names.
- It supports a single series of transactions that combine work spanning its producers and consumers into atomic units.
- It defines a serial order for the messages it consumes and the messages it produces.
- It retains messages it consumes until they have been acknowledged.
- It serializes execution of message listeners registered with its message consumers.

A session can create and service multiple message producers and consumers.

One typical use is to have a thread block on a synchronous **MessageConsumer** (p. 2080) until a message arrives. The thread may then use one or more of the Session's **MessageProducers**.

If a client desires to have one thread produce messages while others consume them, the client should use a separate session for its producing thread.

Certain rules apply to a session's **close** method and are detailed below.

- There is no need to close the producers and consumers of a closed session.
- The close call will block until a receive call or message listener in progress has completed. A blocked message consumer receive call returns null when this session is closed.
- Closing a transacted session must roll back the transaction in progress.
- The close method is the only **Session** (p. 2663) method that can be called concurrently.
- Invoking any other **Session** (p. 2663) method on a closed session must throw an **IllegalStateException** (p. 1621). Closing a closed session must not throw any exceptions.

Transacted Sessions

When a **Session** (p. 2663) is created it can be set to operate in a Transaction based mode. Each **Session** (p. 2663) then operates in a single transaction for all Producers and Consumers of that **Session** (p. 2663). Messages sent and received within a Transaction are grouped into an atomic unit that is committed or rolled back together.

For a **MessageProducer** (p. 2189) this implies that all messages sent by the producer are not sent to the Provider unit the commit call is made. Rolling back the Transaction results in all produced Messages being dropped.

For a **MessageConsumer** (p. 2080) this implies that all received messages are not Acknowledged until the Commit call is made. Rolling back the Transaction results in all Consumed **Message** (p. 2036) being redelivered to the client, the Provider may allow configuration that limits the Maximum number of redeliveries for a **Message** (p. 2036).

Since

1.0

6.603.2 Member Enumeration Documentation

6.603.2.1 enum cms::Session::AcknowledgeMode

Enumerator:

AUTO_ACKNOWLEDGE With this acknowledgment mode, the session automatically acknowledges a client's receipt of a message either when the session has successfully returned from a call to receive or when the message listener the session has called to process the message successfully returns.

DUPS_OK_ACKNOWLEDGE With this acknowledgment mode, the session automatically acknowledges a client's receipt of a message either when the session has successfully returned from a call to receive or when the message listener the session has called to process the message successfully returns. Acknowledgments may be delayed in this mode to increase performance at the cost of the message being redelivered this client fails.

CLIENT_ACKNOWLEDGE With this acknowledgment mode, the client acknowledges a consumed message by calling the message's acknowledge method.

SESSION_TRANSACTED Messages will be consumed when the transaction commits.

INDIVIDUAL_ACKNOWLEDGE **Message** (p. 2036) will be acknowledged individually. Normally the acks sent acknowledge the given message and all messages received before it, this mode only acknowledges one message.

6.603.3 Constructor & Destructor Documentation

6.603.3.1 virtual cms::Session::~~Session () [inline, virtual]

6.603.4 Member Function Documentation

6.603.4.1 virtual void cms::Session::close () throw (CMSEException) [pure virtual]

Closes this session as well as any active child consumers or producers.

Exceptions

CMSEException (p. 960) - If an internal error occurs.

Implements cms::Closeable (p. 952).

Implemented in activemq::cmsutil::PooledSession (p. 2354).

6.603.4.2 virtual void cms::Session::commit () throw (CMSEException) [pure virtual]

Commits all messages done in this transaction and releases any locks currently held.

Exceptions

CMSEException (p. 960) - If an internal error occurs.

IllegalStateException (p. 1621) - if the method is not called by a transacted session.

Implemented in activemq::cmsutil::PooledSession (p. 2354).

Referenced by activemq::cmsutil::PooledSession::commit().

6.603.4.3 virtual QueueBrowser* cms::Session::createBrowser (const cms::Queue * *queue*) throw (CMSEException) [pure virtual]

Creates a new QueueBrowser (p. 2511) to peek at Messages on the given Queue (p. 2510).

Parameters

queue the Queue (p. 2510) to browse

Returns

New QueueBrowser (p. 2511) that is owned by the caller.

Exceptions

CMSEException (p. 960) - If an internal error occurs.

InvalidDestinationException (p. 1697) - if the destination given is invalid.

Implemented in activemq::cmsutil::PooledSession (p. 2354).

6.603.4.4 `virtual QueueBrowser* cms::Session::createBrowser (const cms::Queue * queue, const std::string & selector) throw (CMSException) [pure virtual]`

Creates a new **QueueBrowser** (p. 2511) to peek at Messages on the given **Queue** (p. 2510).

Parameters

queue the **Queue** (p. 2510) to browse

selector the **Message** (p. 2036) selector to filter which messages are browsed.

Returns

New **QueueBrowser** (p. 2511) that is owned by the caller.

Exceptions

CMSException (p. 960) - If an internal error occurs.

InvalidDestinationException (p. 1697) - if the destination given is invalid.

Implemented in **activemq::cmsutil::PooledSession** (p. 2355).

6.603.4.5 `virtual BytesMessage* cms::Session::createBytesMessage () throw (CMSException) [pure virtual]`

Creates a **BytesMessage** (p. 874).

Exceptions

CMSException (p. 960) - If an internal error occurs.

Implemented in **activemq::cmsutil::PooledSession** (p. 2355).

Referenced by **activemq::cmsutil::PooledSession::createBytesMessage()**.

6.603.4.6 `virtual BytesMessage* cms::Session::createBytesMessage (const unsigned char * bytes, std::size_t bytesSize) throw (CMSException) [pure virtual]`

Creates a **BytesMessage** (p. 874) and sets the payload to the passed value.

Parameters

bytes an array of bytes to set in the message

bytesSize the size of the bytes array, or number of bytes to use

Exceptions

CMSException (p. 960) - If an internal error occurs.

Implemented in **activemq::cmsutil::PooledSession** (p. 2355).

6.603.4.7 `virtual MessageConsumer* cms::Session::createConsumer (const Destination * destination) throw (CMSEException) [pure virtual]`

Creates a **MessageConsumer** (p. 2080) for the specified destination.

Parameters

destination the **Destination** (p. 1387) that this consumer receiving messages for.

Returns

pointer to a new **MessageConsumer** (p. 2080) that is owned by the caller (caller deletes)

Exceptions

CMSEException (p. 960) - If an internal error occurs.

InvalidDestinationException (p. 1697) - if an invalid destination is specified.

Implemented in **activemq::cmsutil::PooledSession** (p. 2358).

Referenced by **activemq::cmsutil::PooledSession::createConsumer()**.

6.603.4.8 `virtual MessageConsumer* cms::Session::createConsumer (const Destination * destination, const std::string & selector) throw (CMSEException) [pure virtual]`

Creates a **MessageConsumer** (p. 2080) for the specified destination, using a message selector.

Parameters

destination the **Destination** (p. 1387) that this consumer receiving messages for.

selector the **Message** (p. 2036) Selector to use

Returns

pointer to a new **MessageConsumer** (p. 2080) that is owned by the caller (caller deletes)

Exceptions

CMSEException (p. 960) - If an internal error occurs.

InvalidDestinationException (p. 1697) - if an invalid destination is specified.

InvalidSelectorException (p. 1703) - if the message selector is invalid.

Implemented in **activemq::cmsutil::PooledSession** (p. 2357).

6.603.4.9 `virtual MessageConsumer* cms::Session::createConsumer (const Destination * destination, const std::string & selector, bool noLocal) throw (CMSEException) [pure virtual]`

Creates a **MessageConsumer** (p. 2080) for the specified destination, using a message selector.

Parameters

destination the **Destination** (p. 1387) that this consumer receiving messages for.

selector the **Message** (p. 2036) Selector to use

noLocal if true, and the destination is a topic, inhibits the delivery of messages published by its own connection. The behavior for NoLocal is not specified if the destination is a queue.

Returns

pointer to a new **MessageConsumer** (p. 2080) that is owned by the caller (caller deletes)

Exceptions

CMSEException (p. 960) - If an internal error occurs.

InvalidDestinationException (p. 1697) - if an invalid destination is specified.

InvalidSelectorException (p. 1703) - if the message selector is invalid.

Implemented in **activemq::cmsutil::PooledSession** (p. 2357).

```
6.603.4.10 virtual MessageConsumer* cms::Session::createDurableConsumer (
    const Topic * destination, const std::string & name, const std::string
    & selector, bool noLocal = false ) throw ( CMSEException ) [pure
    virtual]
```

Creates a durable subscriber to the specified topic, using a **Message** (p. 2036) selector.

Sessions that create durable consumers must use the same client Id as was used the last time the subscription was created in order to receive all messages that were delivered while the client was offline.

Parameters

destination the topic to subscribe to

name The name used to identify the subscription

selector the **Message** (p. 2036) Selector to use

noLocal if true, and the destination is a topic, inhibits the delivery of messages published by its own connection. The behavior for NoLocal is not specified if the destination is a queue.

Returns

pointer to a new durable **MessageConsumer** (p. 2080) that is owned by the caller (caller deletes)

Exceptions

CMSEException (p. 960) - If an internal error occurs.

InvalidDestinationException (p. 1697) - if an invalid destination is specified.

InvalidSelectorException (p. 1703) - if the message selector is invalid.

Implemented in **activemq::cmsutil::PooledSession** (p. 2358).

Referenced by **activemq::cmsutil::PooledSession::createDurableConsumer()**.

6.603.4.11 `virtual MapMessage* cms::Session::createMapMessage () throw (CMSEException) [pure virtual]`

Creates a new **MapMessage** (p. 1982).

Exceptions

CMSEException (p. 960) - If an internal error occurs.

Implemented in **activemq::cmsutil::PooledSession** (p. 2359).

Referenced by **activemq::cmsutil::PooledSession::createMapMessage()**.

6.603.4.12 `virtual Message* cms::Session::createMessage () throw (CMSEException) [pure virtual]`

Creates a new **Message** (p. 2036).

Exceptions

CMSEException (p. 960) - If an internal error occurs.

Implemented in **activemq::cmsutil::PooledSession** (p. 2359).

Referenced by **activemq::cmsutil::PooledSession::createMessage()**.

6.603.4.13 `virtual MessageProducer* cms::Session::createProducer (const Destination * destination) throw (CMSEException) [pure virtual]`

Creates a **MessageProducer** (p. 2189) to send messages to the specified destination.

Parameters

destination the **Destination** (p. 1387) to send on

Returns

New **MessageProducer** (p. 2189) that is owned by the caller.

Exceptions

CMSEException (p. 960) - If an internal error occurs.

InvalidDestinationException (p. 1697) - if an invalid destination is specified.

Implemented in **activemq::cmsutil::PooledSession** (p. 2359).

Referenced by **activemq::cmsutil::PooledSession::createProducer()**.

6.603.4.14 `virtual Queue* cms::Session::createQueue (const std::string & queueName) throw (CMSEException) [pure virtual]`

Creates a queue identity given a **Queue** (p. 2510) name.

Parameters

queueName the name of the new **Queue** (p. 2510)

Returns

new **Queue** (p. 2510) pointer that is owned by the caller.

Exceptions

CMSEException (p. 960) - If an internal error occurs.

Implemented in **activemq::cmsutil::PooledSession** (p. 2359).

Referenced by **activemq::cmsutil::PooledSession::createQueue()**.

6.603.4.15 `virtual StreamMessage* cms::Session::createStreamMessage () throw (CMSEException) [pure virtual]`

Creates a new **StreamMessage** (p. 2892).

Exceptions

CMSEException (p. 960) - If an internal error occurs.

Implemented in **activemq::cmsutil::PooledSession** (p. 2360).

Referenced by **activemq::cmsutil::PooledSession::createStreamMessage()**.

6.603.4.16 `virtual TemporaryQueue* cms::Session::createTemporaryQueue () throw (CMSEException) [pure virtual]`

Creates a **TemporaryQueue** (p. 2971) object.

Returns

new **TemporaryQueue** (p. 2971) pointer that is owned by the caller.

Exceptions

CMSEException (p. 960) - If an internal error occurs.

Implemented in **activemq::cmsutil::PooledSession** (p. 2360).

Referenced by **activemq::cmsutil::PooledSession::createTemporaryQueue()**.

6.603.4.17 `virtual TemporaryTopic* cms::Session::createTemporaryTopic () throw (CMSEException) [pure virtual]`

Creates a **TemporaryTopic** (p. 2973) object.

Exceptions

CMSEException (p. 960) - If an internal error occurs.

Implemented in **activemq::cmsutil::PooledSession** (p. 2360).

Referenced by **activemq::cmsutil::PooledSession::createTemporaryTopic()**.

6.603.4.18 `virtual TextMessage* cms::Session::createTextMessage (const std::string & text) throw (CMSEException) [pure virtual]`

Creates a new **TextMessage** (p. 2974) and set the text to the value given.

Parameters

text the initial text for the message

Exceptions

CMSEException (p. 960) - If an internal error occurs.

Implemented in **activemq::cmsutil::PooledSession** (p. 2361).

6.603.4.19 `virtual TextMessage* cms::Session::createTextMessage () throw (CMSEException) [pure virtual]`

Creates a new **TextMessage** (p. 2974).

Exceptions

CMSEException (p. 960) - If an internal error occurs.

Implemented in **activemq::cmsutil::PooledSession** (p. 2361).

Referenced by **activemq::cmsutil::PooledSession::createTextMessage()**.

6.603.4.20 `virtual Topic* cms::Session::createTopic (const std::string & topicName) throw (CMSEException) [pure virtual]`

Creates a topic identity given a **Queue** (p. 2510) name.

Parameters

topicName the name of the new **Topic** (p. 3014)

Returns

new **Topic** (p. 3014) pointer that is owned by the caller.

Exceptions

CMSEException (p. 960) - If an internal error occurs.

Implemented in **activemq::cmsutil::PooledSession** (p. 2361).

Referenced by **activemq::cmsutil::PooledSession::createTopic()**.

6.603.4.21 `virtual AcknowledgeMode cms::Session::getAcknowledgeMode () const throw (CMSEException) [pure virtual]`

Returns the acknowledgment mode of the session.

Returns

the Sessions Acknowledge Mode

Exceptions

CMSEException (p. 960) - If an internal error occurs.

Implemented in **activemq::cmsutil::PooledSession** (p. 2362).

Referenced by **activemq::cmsutil::PooledSession::getAcknowledgeMode()**.

6.603.4.22 **virtual bool cms::Session::isTransacted () const throw (CMSEException)** [pure virtual]

Gets if the Sessions is a Transacted **Session** (p. 2663).

Returns

transacted true - false.

Exceptions

CMSEException (p. 960) - If an internal error occurs.

Implemented in **activemq::cmsutil::PooledSession** (p. 2362).

Referenced by **activemq::cmsutil::PooledSession::isTransacted()**.

6.603.4.23 **virtual void cms::Session::recover () throw (CMSEException)** [pure virtual]

Stops message delivery in this session, and restarts message delivery with the oldest unacknowledged message.

All consumers deliver messages in a serial order. Acknowledging a received message automatically acknowledges all messages that have been delivered to the client.

Restarting a session causes it to take the following actions:

- Stop message delivery
- Mark all messages that might have been delivered but not acknowledged as "redelivered"
- Restart the delivery sequence including all unacknowledged messages that had been previously delivered. Redelivered messages do not have to be delivered in exactly their original delivery order.

Exceptions

CMSEException (p. 960) - if the CMS provider fails to stop and restart message delivery due to some internal error.

IllegalStateException (p. 1621) - if the method is called by a transacted session.

Implemented in **activemq::cmsutil::PooledSession** (p. 2363).

Referenced by **activemq::cmsutil::PooledSession::recover()**.

6.603.4.24 `virtual void cms::Session::rollback () throw (CMSEException)` [pure virtual]

Rolls back all messages done in this transaction and releases any locks currently held.

Exceptions

CMSEException (p. 960) - If an internal error occurs.

IllegalStateException (p. 1621) - if the method is not called by a transacted session.

Implemented in **activemq::cmsutil::PooledSession** (p. 2363).

Referenced by **activemq::cmsutil::PooledSession::rollback()**.

6.603.4.25 `virtual void cms::Session::unsubscribe (const std::string & name) throw (CMSEException)` [pure virtual]

Unsubscribes a durable subscription that has been created by a client.

This method deletes the state being maintained on behalf of the subscriber by its provider. It is erroneous for a client to delete a durable subscription while there is an active **MessageConsumer** (p. 2080) or Subscriber for the subscription, or while a consumed message is part of a pending transaction or has not been acknowledged in the session.

Parameters

name The name used to identify this subscription

Exceptions

CMSEException (p. 960) - If an internal error occurs.

Implemented in **activemq::cmsutil::PooledSession** (p. 2363).

Referenced by **activemq::cmsutil::PooledSession::unsubscribe()**.

The documentation for this class was generated from the following file:

- `src/main/cms/Session.h`

6.604 activemq::cmsutil::SessionCallback Class Reference

Callback for executing any number of operations on a provided CMS Session.

```
#include <src/main/activemq/cmsutil/SessionCallback.h>
```

Inheritance diagram for **activemq::cmsutil::SessionCallback**:

Public Member Functions

- `virtual ~SessionCallback ()`
- `virtual void doInCms (cms::Session *session)=0 throw (cms::CMSEException)`

Execute any number of operations against the supplied CMS session.

6.604.1 Detailed Description

Callback for executing any number of operations on a provided CMS Session.

6.604.2 Constructor & Destructor Documentation

6.604.2.1 `virtual activemq::cmsutil::SessionCallback::~SessionCallback ()`
[inline, virtual]

6.604.3 Member Function Documentation

6.604.3.1 `virtual void activemq::cmsutil::SessionCallback::doInCms (cms::Session * session) throw (cms::CMSException)` [pure virtual]

Execute any number of operations against the supplied CMS session.

Parameters

session the CMS Session

Exceptions

cms::CMSException (p. 960) if thrown by CMS API methods

Implemented in `activemq::cmsutil::CmsTemplate::ProducerExecutor` (p. 2445), and `activemq::cmsutil::CmsTemplate::ReceiveExecutor` (p. 2525).

The documentation for this class was generated from the following file:

- `src/main/activemq/cmsutil/SessionCallback.h`

6.605 activemq::commands::SessionId Class Reference

```
#include <src/main/activemq/commands/SessionId.h>
```

Inheritance diagram for `activemq::commands::SessionId`:

Public Types

- `typedef decaf::lang::PointerComparator< SessionId > COMPARATOR`

Public Member Functions

- `SessionId ()`
- `SessionId (const SessionId &other)`
- `SessionId (const ConnectionId *connectionId, long long sessionId)`
- `SessionId (const ProducerId *producerId)`
- `SessionId (const ConsumerId *consumerId)`
- `virtual ~SessionId ()`

- virtual unsigned char **getDataStructureType** () const

Get the unique identifier that this object and its own Marshaler share.

- virtual **SessionId** * **cloneDataStructure** () const

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)

Copy the contents of the passed object into this object's members, overwriting any existing data.

- virtual std::string **toString** () const

*Returns a string containing the information for this **DataStructure** (p. 1372) such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** *value) const

*Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent.*

- const **Pointer**< **ConnectionId** > & **getParentId** () const
- virtual const std::string & **getConnectionId** () const
- virtual std::string & **getConnectionId** ()
- virtual void **setConnectionId** (const std::string &connectionId)
- virtual long long **getValue** () const
- virtual void **setValue** (long long value)
- virtual int **compareTo** (const **SessionId** &value) const
- virtual bool **equals** (const **SessionId** &value) const
- virtual bool **operator==** (const **SessionId** &value) const
- virtual bool **operator<** (const **SessionId** &value) const
- **SessionId** & **operator=** (const **SessionId** &other)

Static Public Attributes

- static const unsigned char **ID_SESSIONID** = 121

Protected Attributes

- std::string **connectionId**
- long long **value**

6.605.1 Member Typedef Documentation

6.605.1.1 `typedef decaf::lang::PointerComparator<SessionId>
activemq::commands::SessionId::COMPARATOR`

6.605.2 Constructor & Destructor Documentation

6.605.2.1 `activemq::commands::SessionId::SessionId ()`

6.605.2.2 `activemq::commands::SessionId::SessionId (const SessionId & other)`

6.605.2.3 `activemq::commands::SessionId::SessionId (const ConnectionId *
connectionId, long long sessionId)`

6.605.2.4 `activemq::commands::SessionId::SessionId (const ProducerId *
producerId)`

6.605.2.5 `activemq::commands::SessionId::SessionId (const ConsumerId *
consumerId)`

6.605.2.6 `virtual activemq::commands::SessionId::~~SessionId () [virtual]`

6.605.3 Member Function Documentation

6.605.3.1 `virtual SessionId* activemq::commands::SessionId::cloneDataStructure () const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

6.605.3.2 `virtual int activemq::commands::SessionId::compareTo (const SessionId
& value) const [virtual]`

6.605.3.3 `virtual void activemq::commands::SessionId::copyDataStructure (const
DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

6.605.3.4 `virtual bool activemq::commands::SessionId::equals (const
DataStructure * value) const [virtual]`

Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

6.605.3.5 `virtual bool activemq::commands::SessionId::equals (const SessionId & value) const` [virtual]

6.605.3.6 `virtual std::string& activemq::commands::SessionId::getConnectionId ()` [virtual]

6.605.3.7 `virtual const std::string& activemq::commands::SessionId::getConnectionId () const` [virtual]

Referenced by `activemq::commands::ConsumerId::ConsumerId()`, and `activemq::commands::ProducerId::ProducerId()`.

6.605.3.8 `virtual unsigned char activemq::commands::SessionId::getDataStructureType () const` [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataStructure** (p. 1372) type copy.

6.605.3.9 `const Pointer<ConnectionId>& activemq::commands::SessionId::getParentId () const`

6.605.3.10 `virtual long long activemq::commands::SessionId::getValue () const` [virtual]

Referenced by `activemq::commands::ConsumerId::ConsumerId()`, and `activemq::commands::ProducerId::ProducerId()`.

- 6.605.3.11 `virtual bool activemq::commands::SessionId::operator< (const SessionId & value) const` [virtual]
- 6.605.3.12 `SessionId& activemq::commands::SessionId::operator= (const SessionId & other)`
- 6.605.3.13 `virtual bool activemq::commands::SessionId::operator== (const SessionId & value) const` [virtual]
- 6.605.3.14 `virtual void activemq::commands::SessionId::setConnectionId (const std::string & connectionId)` [virtual]
- 6.605.3.15 `virtual void activemq::commands::SessionId::setValue (long long value)` [virtual]
- 6.605.3.16 `virtual std::string activemq::commands::SessionId::toString () const` [virtual]

Returns a string containing the information for this **DataStructure** (p.1372) such as its type and value of its elements.

Returns

formatted string useful for debugging.

6.605.4 Field Documentation

- 6.605.4.1 `std::string activemq::commands::SessionId::connectionId` [protected]
- 6.605.4.2 `const unsigned char activemq::commands::SessionId::ID_SESSIONID = 121` [static]

Referenced by `activemq::state::CommandVisitorAdapter::processRemoveInfo()`.

- 6.605.4.3 `long long activemq::commands::SessionId::value` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/SessionId.h`

6.606 `activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller` Class Reference

Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 2681).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/SessionIdMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller`:

Public Member Functions

- **SessionIdMarshaller** ()
- virtual **~SessionIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.606.1 Detailed Description

Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 2681). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.606.2 Constructor & Destructor Documentation

6.606.2.1 `activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller::SessionIdMarshaller () [inline]`

6.606.2.2 `virtual activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller::~~SessionIdMarshaller () [inline, virtual]`

6.606.3 Member Function Documentation

6.606.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.606.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.606.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1342).

6.606.3.4 virtual void activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1348).

6.606.3.5 virtual int activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1354).

6.606.3.6 virtual void activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1360).

6.606.3.7 virtual void **activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller::tightUnmarshal**
 (**OpenWireFormat** * *wireFormat*, **commands::DataStructure**
 * *dataStructure*, **decaf::io::DataInputStream** * *dataIn*,
utils::BooleanStream * *bs*) throw (**decaf::io::IOException**) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1366).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/SessionIdMarshaller.h`

6.607 **activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller** Class Reference

Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 2685).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/SessionIdMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller**:

Public Member Functions

- **SessionIdMarshaller** ()
- virtual **~SessionIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.607.1 Detailed Description

Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 2685). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.607.2 Constructor & Destructor Documentation

6.607.2.1 `activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller::SessionIdMarshaller () [inline]`

6.607.2.2 `virtual activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller::~~SessionIdMarshaller () [inline, virtual]`

6.607.3 Member Function Documentation

6.607.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.607.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.607.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1342).

6.607.3.4 virtual void activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1348).

6.607.3.5 virtual int activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1354).

6.607.3.6 virtual void activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1360).

6.607.3.7 virtual void **activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller::tightUnmarshal**
 (**OpenWireFormat** * *wireFormat*, **commands::DataStructure**
 * *dataStructure*, **decaf::io::DataInputStream** * *dataIn*,
utils::BooleanStream * *bs*) throw (**decaf::io::IOException**) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1366).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/SessionIdMarshaller.h`

6.608 **activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller** Class Reference

Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 2689).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/SessionIdMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller**:

Public Member Functions

- **SessionIdMarshaller** ()
- virtual **~SessionIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.608.1 Detailed Description

Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 2689). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.608.2 Constructor & Destructor Documentation

6.608.2.1 `activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller::SessionIdMarshaller () [inline]`

6.608.2.2 `virtual activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller::~~SessionIdMarshaller () [inline, virtual]`

6.608.3 Member Function Documentation

6.608.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.608.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.608.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1342).

6.608.3.4 virtual void activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1348).

6.608.3.5 virtual int activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1354).

6.608.3.6 virtual void activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1360).

6.608.3.7 virtual void **activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller::tightUnmarshal**
 (**OpenWireFormat** * *wireFormat*, **commands::DataStructure**
 * *dataStructure*, **decaf::io::DataInputStream** * *dataIn*,
utils::BooleanStream * *bs*) throw (**decaf::io::IOException**) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1366).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/SessionIdMarshaller.h`

6.609 **activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller** Class Reference

Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 2693).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/SessionIdMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller**:

Public Member Functions

- **SessionIdMarshaller** ()
- virtual **~SessionIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.609.1 Detailed Description

Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 2693). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.609.2 Constructor & Destructor Documentation

6.609.2.1 `activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller::SessionIdMarshaller () [inline]`

6.609.2.2 `virtual activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller::~~SessionIdMarshaller () [inline, virtual]`

6.609.3 Member Function Documentation

6.609.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.609.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.609.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1342).

6.609.3.4 virtual void activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1348).

6.609.3.5 virtual int activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1354).

6.609.3.6 virtual void activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1360).

6.609.3.7 virtual void **activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller::tightUnmarshal**
 (**OpenWireFormat** * *wireFormat*, **commands::DataStructure**
 * *dataStructure*, **decaf::io::DataInputStream** * *dataIn*,
utils::BooleanStream * *bs*) throw (**decaf::io::IOException**) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1366).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/SessionIdMarshaller.h`

6.610 **activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller** Class Reference

Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 2697).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/SessionIdMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller**:

Public Member Functions

- **SessionIdMarshaller** ()
- virtual **~SessionIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.610.1 Detailed Description

Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 2697). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.610.2 Constructor & Destructor Documentation

6.610.2.1 `activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller::SessionIdMarshaller () [inline]`

6.610.2.2 `virtual activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller::~~SessionIdMarshaller () [inline, virtual]`

6.610.3 Member Function Documentation

6.610.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.610.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.610.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1342).

6.610.3.4 virtual void activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1348).

6.610.3.5 virtual int activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1354).

6.610.3.6 virtual void activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1360).

```
6.610.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::SessionIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1366).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/SessionIdMarshaller.h

6.611 activemq::commands::SessionInfo Class Reference

```
#include <src/main/activemq/commands/SessionInfo.h>
```

Inheritance diagram for **activemq::commands::SessionInfo**:

Public Member Functions

- **SessionInfo** ()
- virtual **~SessionInfo** ()
- virtual unsigned char **getDataStructureType** () const

Get the unique identifier that this object and its own Marshaler share.

- virtual **SessionInfo** * **cloneDataStructure** () const

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)

Copy the contents of the passed object into this object's members, overwriting any existing data.

- virtual std::string **toString** () const

*Returns a string containing the information for this **DataStructure** (p. 1372) such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** *value) const

*Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent.*

- unsigned int **getAckMode** () const

- void **setAckMode** (unsigned int mode)

- virtual const **Pointer**< **SessionId** > & **getSessionId** () const

- virtual **Pointer**< **SessionId** > & **getSessionId** ()

- virtual void **setSessionId** (const **Pointer**< **SessionId** > &sessionId)

- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor *visitor) throw (exceptions::ActiveMQException)

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_SESSIONINFO** = 4

Protected Member Functions

- **SessionInfo** (const **SessionInfo** &)
- **SessionInfo** & **operator=** (const **SessionInfo** &)

Protected Attributes

- **Pointer**< **SessionId** > **sessionId**

6.611.1 Constructor & Destructor Documentation

6.611.1.1 `activemq::commands::SessionInfo::SessionInfo (const SessionInfo &)`
[inline, protected]

6.611.1.2 `activemq::commands::SessionInfo::SessionInfo ()`

6.611.1.3 `virtual activemq::commands::SessionInfo::~~SessionInfo ()` [virtual]

6.611.2 Member Function Documentation

6.611.2.1 `virtual SessionInfo* activemq::commands::SessionInfo::cloneDataStructure ()`
`const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1373).

6.611.2.2 `virtual void activemq::commands::SessionInfo::copyDataStructure (const DataStructure * src)` [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

Reimplemented from `activemq::commands::BaseCommand` (p. 598).

6.611.2.3 `virtual bool activemq::commands::SessionInfo::equals (const DataStructure * value) const` [virtual]

Compares the `DataStructure` (p. 1372) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 598).

6.611.2.4 `unsigned int activemq::commands::SessionInfo::getAckMode () const`
[inline]

6.611.2.5 `virtual unsigned char activemq::commands::SessionInfo::getDataStructureType () const` [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataStructure** (p. 1372) type copy.

Implements **activemq::commands::DataStructure** (p. 1375).

6.611.2.6 `virtual const Pointer<SessionId>& activemq::commands::SessionInfo::getSessionId () const`
[virtual]

6.611.2.7 `virtual Pointer<SessionId>& activemq::commands::SessionInfo::getSessionId ()`
[virtual]

6.611.2.8 `SessionInfo& activemq::commands::SessionInfo::operator= (const SessionInfo &)` [inline, protected]

6.611.2.9 `void activemq::commands::SessionInfo::setAckMode (unsigned int mode)` [inline]

6.611.2.10 `virtual void activemq::commands::SessionInfo::setSessionId (const Pointer< SessionId > & sessionId)` [virtual]

6.611.2.11 `virtual std::string activemq::commands::SessionInfo::toString () const`
[virtual]

Returns a string containing the information for this **DataStructure** (p. 1372) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 602).

6.611.2.12 `virtual Pointer<Command> activemq::commands::SessionInfo::visit (activemq::state::CommandVisitor * visitor) throw (exceptions::ActiveMQException)` [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

Returns

a **Response** (p. 2611) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 995).

6.611.3 Field Documentation

6.611.3.1 `const unsigned char activemq::commands::SessionInfo::ID_SESSIONINFO = 4` [static]

6.611.3.2 `Pointer<SessionId> activemq::commands::SessionInfo::sessionId` [protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/SessionInfo.h`

6.612 **activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller** Class Reference

Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 2705).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/SessionInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller**:

Public Member Functions

- **SessionInfoMarshaller** ()
- virtual **~SessionInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.612.1 Detailed Description

Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p.2705). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.612.2 Constructor & Destructor Documentation

6.612.2.1 **activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller::SessionInfoMarshaller**
() [inline]

6.612.2.2 **virtual**
activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller::~~SessionInfoMarshaller
() [inline, virtual]

6.612.3 Member Function Documentation

6.612.3.1 **virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller::createObject**
() const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1331).

6.612.3.2 **virtual unsigned char activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller::getDataStructureType**
() const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.612.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 631).

6.612.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 632).

6.612.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 633).

```
6.612.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 634).

```
6.612.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 635).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/SessionInfoMarshaller.h`

6.613 activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 2709).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/SessionInfoMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller`:

Public Member Functions

- **SessionInfoMarshaller** ()
- virtual **~SessionInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.613.1 Detailed Description

Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p.2709). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.613.2 Constructor & Destructor Documentation

6.613.2.1 activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller::SessionInfoMarshaller () [inline]

6.613.2.2 virtual activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller::~~SessionInfoMarshaller () [inline, virtual]

6.613.3 Member Function Documentation

6.613.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1331).

6.613.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1336).

6.613.3.3 virtual void activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 611).

6.613.3.4 virtual void activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 612).

6.613.3.5 virtual int activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 613).

6.613.3.6 virtual void **activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller::tightMarshal2**
(**OpenWireFormat** * *wireFormat*, **commands::DataStructure**
* *dataStructure*, **decaf::io::DataOutputStream** * *dataOut*,
utils::BooleanStream * *bs*) throw (**decaf::io::IOException**) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 614).

6.613.3.7 virtual void **activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller::tightUnmarshal**
(**OpenWireFormat** * *wireFormat*, **commands::DataStructure**
* *dataStructure*, **decaf::io::DataInputStream** * *dataIn*,
utils::BooleanStream * *bs*) throw (**decaf::io::IOException**) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 615).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/SessionInfoMarshaller.h`

6.614 activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 2713).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/SessionInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller**:

Public Member Functions

- **SessionInfoMarshaller** ()
- virtual **~SessionInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.614.1 Detailed Description

Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p.2713). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.614.2 Constructor & Destructor Documentation

6.614.2.1 activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller::SessionInfoMarshaller () [inline]

6.614.2.2 virtual activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller::~~SessionInfoMarshaller () [inline, virtual]

6.614.3 Member Function Documentation

6.614.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1331).

6.614.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1336).

6.614.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 618).

6.614.3.4 `virtual void activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 619).

6.614.3.5 `virtual int activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 620).

```
6.614.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 621).

```
6.614.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 622).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/SessionInfoMarshaller.h`

6.615 activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 2717).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/SessionInfoMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller`:

Public Member Functions

- **SessionInfoMarshaller** ()
- virtual **~SessionInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.615.1 Detailed Description

Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p.2717). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.615.2 Constructor & Destructor Documentation

6.615.2.1 activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller::SessionInfoMarshaller () [inline]

6.615.2.2 virtual
activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller::~~SessionInfoMarshaller () [inline, virtual]

6.615.3 Member Function Documentation

6.615.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1331).

6.615.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1336).

```

6.615.3.3  virtual void ac-
            tivemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller::looseMarshal
            ( OpenWireFormat * wireFormat, commands::DataStructure *
              dataStructure, decaf::io::DataOutputStream * dataOut ) throw (
              decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 624).

```

6.615.3.4  virtual void ac-
            tivemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller::looseUnmarshal
            ( OpenWireFormat * wireFormat, commands::DataStructure *
              dataStructure, decaf::io::DataInputStream * dataIn ) throw (
              decaf::io::IOException ) [virtual]

```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 625).

```

6.615.3.5  virtual int ac-
            tivemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller::tightMarshal1
            ( OpenWireFormat * wireFormat, commands::DataStructure
              * dataStructure, utils::BooleanStream * bs ) throw (
              decaf::io::IOException ) [virtual]

```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 626).

```
6.615.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 628).

```
6.615.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 629).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/SessionInfoMarshaller.h`

6.616 activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 2721).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/SessionInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller**:

Public Member Functions

- **SessionInfoMarshaller** ()
- virtual **~SessionInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.616.1 Detailed Description

Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p.2721). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.616.2 Constructor & Destructor Documentation

6.616.2.1 activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller::SessionInfoMarshaller () [inline]

6.616.2.2 virtual activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller::~~SessionInfoMarshaller () [inline, virtual]

6.616.3 Member Function Documentation

6.616.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1331).

6.616.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1336).

6.616.3.3 virtual void activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 604).

6.616.3.4 virtual void activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 605).

6.616.3.5 virtual int activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 606).

6.616.3.6 virtual void **activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller::tightMarshal2**
(**OpenWireFormat** * *wireFormat*, **commands::DataStructure**
* *dataStructure*, **decaf::io::DataOutputStream** * *dataOut*,
utils::BooleanStream * *bs*) throw (**decaf::io::IOException**) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 608).

6.616.3.7 virtual void **activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller::tightUnmarshal**
(**OpenWireFormat** * *wireFormat*, **commands::DataStructure**
* *dataStructure*, **decaf::io::DataInputStream** * *dataIn*,
utils::BooleanStream * *bs*) throw (**decaf::io::IOException**) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 609).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/SessionInfoMarshaller.h`

6.617 activemq::cmsutil::SessionPool Class Reference

A pool of CMS sessions from the same connection and with the same acknowledge mode.

```
#include <src/main/activemq/cmsutil/SessionPool.h>
```

Public Member Functions

- **SessionPool** (`cms::Connection *connection`, `cms::Session::AcknowledgeMode ackMode`, `ResourceLifecycleManager *resourceLifecycleManager`)
Constructs a session pool.
- virtual `~SessionPool ()`
Destroys the pooled session objects, but not the underlying session resources.
- virtual `PooledSession * takeSession ()` throw (`cms::CMSException`)
Takes a session from the pool, creating one if necessary.
- virtual void `returnSession (PooledSession *session)`
Returns a session to the pool.
- `ResourceLifecycleManager * getResourceLifecycleManager ()`

6.617.1 Detailed Description

A pool of CMS sessions from the same connection and with the same acknowledge mode. Internal session resources are managed through a provided `ResourceLifecycleManager` (p. 2608), not by this pool. This class is thread-safe.

6.617.2 Constructor & Destructor Documentation

6.617.2.1 `activemq::cmsutil::SessionPool::SessionPool (cms::Connection * connection, cms::Session::AcknowledgeMode ackMode, ResourceLifecycleManager * resourceLifecycleManager)`

Constructs a session pool.

Parameters

connection the connection to be used for creating all sessions.

ackMode the acknowledge mode to be used for all sessions

resourceLifecycleManager the object responsible for managing the lifecycle of any allocated **cms::Session** (p. 2663) resources.

6.617.2.2 virtual activemq::cmsutil::SessionPool::~SessionPool () [virtual]

Destroys the pooled session objects, but not the underlying session resources.

That is the job of the **ResourceLifecycleManager** (p. 2608).

6.617.3 Member Function Documentation

6.617.3.1 ResourceLifecycleManager* activemq::cmsutil::SessionPool::getResourceLifecycleManager () [inline]

6.617.3.2 virtual void activemq::cmsutil::SessionPool::returnSession (PooledSession * session) [virtual]

Returns a session to the pool.

Parameters

session the session to be returned.

6.617.3.3 virtual PooledSession* activemq::cmsutil::SessionPool::takeSession () throw (cms::CMSException) [virtual]

Takes a session from the pool, creating one if necessary.

Returns

the pooled session object

Exceptions

cms::CMSException (p. 960) if an error occurred

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/SessionPool.h

6.618 activemq::state::SessionState Class Reference

```
#include <src/main/activemq/state/SessionState.h>
```

Public Member Functions

- **SessionState** (const **Pointer**< **SessionInfo** > &info)
- virtual ~**SessionState** ()
- std::string **toString** () const
- const **Pointer**< **SessionInfo** > **getInfo** () const
- void **addProducer** (const **Pointer**< **ProducerInfo** > &info)
- **Pointer**< **ProducerState** > **removeProducer** (const **Pointer**< **ProducerId** > &id)
- void **addConsumer** (const **Pointer**< **ConsumerInfo** > &info)
- **Pointer**< **ConsumerState** > **removeConsumer** (const **Pointer**< **ConsumerId** > &id)
- std::vector< **Pointer**< **ProducerState** > > **getProducerStates** () const
- **Pointer**< **ProducerState** > **getProducerState** (const **Pointer**< **ProducerId** > &id)
- std::vector< **Pointer**< **ConsumerState** > > **getConsumerStates** () const
- **Pointer**< **ConsumerState** > **getConsumerState** (const **Pointer**< **ConsumerId** > &id)
- void **checkShutdown** () const
- void **shutdown** ()

6.618.1 Constructor & Destructor Documentation

6.618.1.1 `activemq::state::SessionState::SessionState (const Pointer< SessionInfo > & info)`

6.618.1.2 `virtual activemq::state::SessionState::~~SessionState () [virtual]`

6.618.2 Member Function Documentation

6.618.2.1 `void activemq::state::SessionState::addConsumer (const Pointer< ConsumerInfo > & info) [inline]`

6.618.2.2 `void activemq::state::SessionState::addProducer (const Pointer< ProducerInfo > & info) [inline]`

6.618.2.3 `void activemq::state::SessionState::checkShutdown () const`

6.618.2.4 `Pointer<ConsumerState> activemq::state::SessionState::getConsumerState (const Pointer< ConsumerId > & id) [inline]`

6.618.2.5 `std::vector< Pointer<ConsumerState> > activemq::state::SessionState::getConsumerStates () const [inline]`

6.618.2.6 `const Pointer<SessionInfo> activemq::state::SessionState::getInfo () const [inline]`

6.618.2.7 `Pointer<ProducerState> activemq::state::SessionState::getProducerState (const Pointer< ProducerId > & id) [inline]`

6.618.2.8 `std::vector< Pointer<ProducerState> > activemq::state::SessionState::getProducerStates () const [inline]`

6.618.2.9 `Pointer<ConsumerState> activemq::state::SessionState::removeConsumer (const Pointer< ConsumerId > & id) [inline]`

6.618.2.10 `Pointer<ProducerState> activemq::state::SessionState::removeProducer (const Pointer< ProducerId > & id) [inline]`

6.618.2.11 `void activemq::state::SessionState::shutdown () [inline]`

6.618.2.12 `std::string activemq::state::SessionState::toString () const`

The documentation for this class was generated from the following file:

- `src/main/activemq/state/SessionState.h`

6.619 decaf::util::Set< E > Class Template Reference

A collection that contains no duplicate elements.

```
#include <src/main/decaf/util/Set.h>
```

Inheritance diagram for decaf::util::Set< E >:

Public Member Functions

- virtual `~Set()`

6.619.1 Detailed Description

```
template<typename E> class decaf::util::Set< E >
```

A collection that contains no duplicate elements. More formally, sets contain no pair of elements `e1` and `e2` such that `e1 == e2`, and at most one null element. As implied by its name, this interface models the mathematical set abstraction.

The additional stipulation on constructors is, not surprisingly, that all constructors must create a set that contains no duplicate elements (as defined above).

Note: Great care must be exercised if mutable objects are used as set elements. The behavior of a set is not specified if the value of an object is changed in a manner that affects equals comparisons while the object is an element in the set.

Since

1.0

6.619.2 Constructor & Destructor Documentation

6.619.2.1 `template<typename E> virtual decaf::util::Set< E >::~~Set ()`
[inline, virtual]

The documentation for this class was generated from the following file:

- `src/main/decaf/util/Set.h`

6.620 decaf::lang::Short Class Reference

```
#include <src/main/decaf/lang/Short.h>
```

Inheritance diagram for decaf::lang::Short:

Public Member Functions

- **Short** (short value)
- **Short** (const std::string &value) throw (exceptions::NumberFormatException)
- virtual ~**Short** ()
- virtual int **compareTo** (const **Short** &s) const
*Compares this **Short** (p. 2729) instance with another.*
- bool **equals** (const **Short** &s) const
- virtual bool **operator==** (const **Short** &s) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const **Short** &s) const
Compares this object to another and returns true if this object is considered to be less than the one passed.
- virtual int **compareTo** (const short &s) const
*Compares this **Short** (p. 2729) instance with another.*
- bool **equals** (const short &s) const
- virtual bool **operator==** (const short &s) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const short &s) const
Compares this object to another and returns true if this object is considered to be less than the one passed.
- std::string **toString** () const
- virtual double **doubleValue** () const
Answers the double value which the receiver represents.
- virtual float **floatValue** () const
Answers the float value which the receiver represents.
- virtual unsigned char **byteValue** () const
Answers the byte value which the receiver represents.
- virtual short **shortValue** () const
Answers the short value which the receiver represents.
- virtual int **intValue** () const
Answers the int value which the receiver represents.
- virtual long long **longValue** () const
Answers the long value which the receiver represents.

Static Public Member Functions

- static std::string **toString** (short value)
- static **Short decode** (const std::string &value) throw (exceptions::NumberFormatException)
*Decodes a String into a **Short** (p. 2729).*
- static short **reverseBytes** (short value)
Returns the value obtained by reversing the order of the bytes in the two's complement representation of the specified short value.
- static short **parseShort** (const std::string &s, int radix) throw (exceptions::NumberFormatException)
Parses the string argument as a signed short in the radix specified by the second argument.
- static short **parseShort** (const std::string &s) throw (exceptions::NumberFormatException)
Parses the string argument as a signed decimal short.
- static **Short valueOf** (short value)
*Returns a **Short** (p. 2729) instance representing the specified short value.*
- static **Short valueOf** (const std::string &value) throw (exceptions::NumberFormatException)
*Returns a **Short** (p. 2729) object holding the value given by the specified std::string.*
- static **Short valueOf** (const std::string &value, int radix) throw (exceptions::NumberFormatException)
*Returns a **Short** (p. 2729) object holding the value extracted from the specified std::string when parsed with the radix given by the second argument.*

Static Public Attributes

- static const int **SIZE** = 16
Size of this objects primitive type in bits.
- static const short **MAX_VALUE** = (short)0x7FFF
Max Value for this Object's primitive type.
- static const short **MIN_VALUE** = (short)0x8000
Max Value for this Object's primitive type.

6.620.1 Constructor & Destructor Documentation

6.620.1.1 decaf::lang::Short::Short (short value)

Parameters

value - short to wrap

6.620.1.2 decaf::lang::Short::Short (const std::string & *value*) throw (exceptions::NumberFormatException)

Parameters

value - string value to convert to short and wrap

Exceptions

NumberFormatException

6.620.1.3 virtual decaf::lang::Short::~~Short () [inline, virtual]

6.620.2 Member Function Documentation

6.620.2.1 virtual unsigned char decaf::lang::Short::byteValue () const [inline, virtual]

Answers the byte value which the receiver represents.

Returns

int the value of the receiver.

6.620.2.2 virtual int decaf::lang::Short::compareTo (const short & *s*) const [virtual]

Compares this **Short** (p. 2729) instance with another.

Parameters

s - the **Short** (p. 2729) instance to be compared

Returns

zero if this object represents the same short value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

Implements **decaf::lang::Comparable**< **short** > (p. 1010).

6.620.2.3 virtual int decaf::lang::Short::compareTo (const Short & *s*) const [virtual]

Compares this **Short** (p. 2729) instance with another.

Parameters

s - the **Short** (p. 2729) instance to be compared

Returns

zero if this object represents the same short value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

6.620.2.4 `static Short decaf::lang::Short::decode (const std::string & value)
throw (exceptions::NumberFormatException) [static]`

Decodes a String into a **Short** (p.2729).

Accepts decimal, hexadecimal, and octal numbers given by the following grammar:

The sequence of characters following an (optional) negative sign and/or radix specifier ("0x", "0X", "#", or leading zero) is parsed as by the **Short.parseShort** (p.2735) method with the indicated radix (10, 16, or 8). This sequence of characters must represent a positive value or a **NumberFormatException** will be thrown. The result is negated if first character of the specified String is the minus sign. No whitespace characters are permitted in the string.

Parameters

value - The string to decode

Returns

a **Short** (p.2729) object containing the decoded value

Exceptions

NumberFormatException if the string is not formatted correctly.

6.620.2.5 `virtual double decaf::lang::Short::doubleValue () const [inline,
virtual]`

Answers the double value which the receiver represents.

Returns

double the value of the receiver.

6.620.2.6 `bool decaf::lang::Short::equals (const Short & s) const [inline]`

Returns

true if the two **Short** (p.2729) Objects have the same value.

6.620.2.7 `bool decaf::lang::Short::equals (const short & s) const [inline,
virtual]`

Returns

true if the two **Short** (p.2729) Objects have the same value.

Implements **decaf::lang::Comparable< short >** (p.1010).

6.620.2.8 `virtual float decaf::lang::Short::floatValue () const [inline, virtual]`

Answers the float value which the receiver represents.

Returns

float the value of the receiver.

6.620.2.9 `virtual int decaf::lang::Short::intValue () const [inline, virtual]`

Answers the int value which the receiver represents.

Returns

int the value of the receiver.

6.620.2.10 `virtual long long decaf::lang::Short::longValue () const [inline, virtual]`

Answers the long value which the receiver represents.

Returns

long the value of the receiver.

6.620.2.11 `virtual bool decaf::lang::Short::operator< (const Short & s) const [inline, virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

Parameters

s - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

6.620.2.12 `virtual bool decaf::lang::Short::operator< (const short & s) const [inline, virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

Parameters

s - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

Implements `decaf::lang::Comparable< short >` (p.1011).

6.620.2.13 `virtual bool decaf::lang::Short::operator==(const Short & s) const`
[inline, virtual]

Compares equality between this object and the one passed.

Parameters

s - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

6.620.2.14 `virtual bool decaf::lang::Short::operator==(const short & s) const`
[inline, virtual]

Compares equality between this object and the one passed.

Parameters

s - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

Implements `decaf::lang::Comparable< short >` (p.1011).

6.620.2.15 `static short decaf::lang::Short::parseShort (const std::string & s, int`
radix) throw (exceptions::NumberFormatException) [static]

Parses the string argument as a signed short in the radix specified by the second argument.

The characters in the string must all be digits, of the specified radix (as determined by whether `Character.digit(char, int)` (p. 917) returns a nonnegative value) except that the first character may be an ASCII minus sign '-' to indicate a negative value. The resulting byte value is returned.

An exception of type `NumberFormatException` is thrown if any of the following situations occurs:
* The first argument is null or is a string of length zero. * The radix is either smaller than `Character.MIN_RADIX` (p. 921) or larger than `Character.MAX_RADIX` (p. 921). * Any character of the string is not a digit of the specified radix, except that the first character may be a minus sign '-' provided that the string is longer than length 1. * The value represented by the string is not a value of type short.

Parameters

s - the String containing the short representation to be parsed

radix - the radix to be used while parsing s

Returns

the short represented by the string argument in the specified radix.

Exceptions

NumberFormatException - If String does not contain a parsable short.

6.620.2.16 `static short decaf::lang::Short::parseShort (const std::string & s)
 throw (exceptions::NumberFormatException) [static]`

Parses the string argument as a signed decimal short.

The characters in the string must all be decimal digits, except that the first character may be an ASCII minus sign '-' to indicate a negative value. The resulting short value is returned, exactly as if the argument and the radix 10 were given as arguments to the `parseShort(const std::string, int)` method.

Parameters

s - String to convert to a short

Returns

the converted short value

Exceptions

NumberFormatException if the string is not a short.

6.620.2.17 `static short decaf::lang::Short::reverseBytes (short value) [static]`

Returns the value obtained by reversing the order of the bytes in the two's complement representation of the specified short value.

Parameters

value - the short whose bytes we are to reverse

Returns

the reversed short.

6.620.2.18 `virtual short decaf::lang::Short::shortValue () const [inline,
 virtual]`

Answers the short value which the receiver represents.

Returns

int the value of the receiver.

6.620.2.19 `static std::string decaf::lang::Short::toString (short value) [static]`

Returns

a string representing the primitive value as Base 10

6.620.2.20 `std::string decaf::lang::Short::toString () const`**Returns**

this **Short** (p. 2729) Object as a String Representation

6.620.2.21 `static Short decaf::lang::Short::valueOf (short value) [static]`

Returns a **Short** (p. 2729) instance representing the specified short value.

Parameters

value - the short to wrap

Returns

the new **Short** (p. 2729) object wrapping value.

6.620.2.22 `static Short decaf::lang::Short::valueOf (const std::string & value)
throw (exceptions::NumberFormatException) [static]`

Returns a **Short** (p. 2729) object holding the value given by the specified std::string.

The argument is interpreted as representing a signed decimal short, exactly as if the argument were given to the `parseShort(std::string)` method. The result is a **Short** (p. 2729) object that represents the short value specified by the string.

Parameters

value - std::string to parse as base 10

Returns

new **Short** (p. 2729) Object wrapping the primitive

Exceptions

NumberFormatException if the string is not a decimal short.

6.620.2.23 `static Short decaf::lang::Short::valueOf (const std::string & value, int
radix) throw (exceptions::NumberFormatException) [static]`

Returns a **Short** (p. 2729) object holding the value extracted from the specified std::string when parsed with the radix given by the second argument.

The first argument is interpreted as representing a signed short in the radix specified by the second argument, exactly as if the argument were given to the `parseShort(std::string, int)` method. The result is a **Short** (p. 2729) object that represents the short value specified by the string.

Parameters

value - std::string to parse as base (radix)

radix - base of the string to parse.

Returns

new **Short** (p. 2729) Object wrapping the primitive

Exceptions

NumberFormatException if the string is not a valid short.

6.620.3 Field Documentation

6.620.3.1 `const short decaf::lang::Short::MAX_VALUE = (short)0x7FFF`
[static]

Max Value for this Object's primitive type.

6.620.3.2 `const short decaf::lang::Short::MIN_VALUE = (short)0x8000` [static]

Max Value for this Object's primitive type.

6.620.3.3 `const int decaf::lang::Short::SIZE = 16` [static]

Size of this objects primitive type in bits.

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**Short.h**

6.621 decaf::internal::nio::ShortArrayBuffer Class Reference

```
#include <src/main/decaf/internal/nio/ShortArrayBuffer.h>
```

Inheritance diagram for decaf::internal::nio::ShortArrayBuffer:

Public Member Functions

- **ShortArrayBuffer** (std::size_t capacity, bool readOnly=false)
*Creates a **ShortArrayBuffer** (p. 2738) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*
- **ShortArrayBuffer** (short *array, std::size_t offset, std::size_t capacity, bool readOnly=false) throw (decaf::lang::exceptions::NullPointerException)
*Creates a **ShortArrayBuffer** (p. 2738) object that wraps the given array.*
- **ShortArrayBuffer** (ByteArrayPerspective &array, std::size_t offset, std::size_t capacity, bool readOnly=false) throw (decaf::lang::exceptions::IndexOutOfBoundsException)
*Creates a byte buffer that wraps the passed **ByteArrayPerspective** (p. 843) and start at the given offset.*
- **ShortArrayBuffer** (const **ShortArrayBuffer** &other)

Create a **ShortArrayBuffer** (p. 2738) that mirrors this one, meaning it shares a reference to this buffers **ByteArrayPerspective** (p. 843) and when changes are made to that data it is reflected in both.

- virtual **~ShortArrayBuffer** ()
- virtual short * **array** () throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException)
Returns the short array that backs this buffer (optional operation).
- virtual std::size_t **arrayOffset** () throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException)
Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).
- virtual ShortBuffer * **asReadOnlyBuffer** () const
Creates a new, read-only short buffer that shares this buffer's content.
- virtual ShortBuffer & **compact** () throw (decaf::nio::ReadOnlyBufferException)
Compacts this buffer.
- virtual ShortBuffer * **duplicate** ()
Creates a new short buffer that shares this buffer's content.
- virtual short **get** () throw (decaf::nio::BufferUnderflowException)
Relative get method.
- virtual short **get** (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)
Absolute get method.
- virtual bool **hasArray** () const
Tells whether or not this buffer is backed by an accessible short array.
- virtual bool **isReadOnly** () const
Tells whether or not this buffer is read-only.
- virtual ShortBuffer & **put** (short value) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException)
Writes the given doubles into this buffer at the current position, and then increments the position.
- virtual ShortBuffer & **put** (std::size_t index, short value) throw (lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException)
Writes the given doubles into this buffer at the given index.
- virtual ShortBuffer * **slice** () const
Creates a new ShortBuffer whose content is a shared subsequence of this buffer's content.

Protected Member Functions

- virtual void **setReadOnly** (bool value)
*Sets this **ByteBuffer** (p. 806) as Read-Only.*

6.621.1 Constructor & Destructor Documentation

6.621.1.1 decaf::internal::nio::ShortArrayBuffer::ShortArrayBuffer (std::size_t capacity, bool readOnly = false)

Creates a **ShortArrayBuffer** (p. 2738) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.

The array is initially created with all elements initialized to zero.

Parameters

capacity - size of the array, this is the limit we read and write to.
readOnly - should this buffer be read-only, default as false

6.621.1.2 decaf::internal::nio::ShortArrayBuffer::ShortArrayBuffer (short * array, std::size_t offset, std::size_t capacity, bool readOnly = false) throw (decaf::lang::exceptions::NullPointerException)

Creates a **ShortArrayBuffer** (p. 2738) object that wraps the given array.

If the own flag is set then it will delete this array when this object is deleted.

Parameters

array - array to wrap
offset - the position that is this buffers start pos.
capacity - size of the array, this is the limit we read and write to.
readOnly - should this buffer be read-only, default as false

Exceptions

NullPointerException if buffer is NULL

6.621.1.3 decaf::internal::nio::ShortArrayBuffer::ShortArrayBuffer (ByteArrayPerspective & array, std::size_t offset, std::size_t capacity, bool readOnly = false) throw (decaf::lang::exceptions::IndexOutOfBoundsException)

Creates a byte buffer that wraps the passed **ByteArrayPerspective** (p. 843) and start at the given offset.

The capacity and limit of the new **ShortArrayBuffer** (p. 2738) will be that of the remaining capacity of the passed buffer.

Parameters

array - the **ByteArrayPerspective** (p. 843) to wrap

offset - the position that is this buffers start pos.

capacity - the length of the array we are wrapping or limit.

readOnly - is this a readOnly buffer.

Exceptions

IndexOutOfBoundsException if offset is greater than array capacity.

6.621.1.4 decaf::internal::nio::ShortArrayBuffer::ShortArrayBuffer (const ShortArrayBuffer & other)

Create a **ShortArrayBuffer** (p. 2738) that mirrors this one, meaning it shares a reference to this buffers **ByteArrayPerspective** (p. 843) and when changes are made to that data it is reflected in both.

Parameters

other - the **ShortArrayBuffer** (p. 2738) this one is to mirror.

6.621.1.5 virtual decaf::internal::nio::ShortArrayBuffer::~ShortArrayBuffer () [virtual]

6.621.2 Member Function Documentation

6.621.2.1 virtual short* decaf::internal::nio::ShortArrayBuffer::array () throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException) [virtual]

Returns the short array that backs this buffer (optional operation).

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

the array that backs this Buffer

Exceptions

ReadOnlyBufferException if this Buffer is read only.

UnsupportedOperationException if the underlying store has no array.

Implements **decaf::nio::ShortBuffer** (p. 2748).

6.621.2.2 virtual std::size_t decaf::internal::nio::ShortArrayBuffer::arrayOffset () throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException) [virtual]

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

The offset into the backing array where index zero starts.

Exceptions

ReadOnlyBufferException if this Buffer is read only.

UnsupportedOperationException if the underlying store has no array.

Implements **decaf::nio::ShortBuffer** (p. 2749).

6.621.2.3 virtual ShortBuffer* decaf::internal::nio::ShortArrayBuffer::asReadOnlyBuffer () const [virtual]

Creates a new, read-only short buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the `duplicate` method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns

The new, read-only short buffer which the caller then owns.

Implements **decaf::nio::ShortBuffer** (p. 2749).

6.621.2.4 virtual ShortBuffer& decaf::internal::nio::ShortArrayBuffer::compact () throw (decaf::nio::ReadOnlyBufferException) [virtual]

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index `p = position()` (p. 746) is copied to index zero, the byte at index `p + 1` is copied to index one, and so forth until the byte at index `limit()` (p. 745) - 1 is copied to index `n = limit()` (p. 745) - 1 - `p`. The buffer's position is then set to `n+1` and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns

a reference to this ShortBuffer

Exceptions

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::ShortBuffer** (p. 2750).

6.621.2.5 `virtual ShortBuffer* decaf::internal::nio::ShortArrayBuffer::duplicate ()` [virtual]

Creates a new short buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

a new short Buffer which the caller owns.

Implements `decaf::nio::ShortBuffer` (p. 2750).

6.621.2.6 `virtual short decaf::internal::nio::ShortArrayBuffer::get ()` throw (`decaf::nio::BufferUnderflowException`) [virtual]

Relative get method.

Reads the value at this buffer's current position, and then increments the position.

Returns

the short at the current position

Exceptions

BufferUnderflowException if there no more data to return

Implements `decaf::nio::ShortBuffer` (p. 2752).

6.621.2.7 `virtual short decaf::internal::nio::ShortArrayBuffer::get (std::size_t index)` const throw (`lang::exceptions::IndexOutOfBoundsException`) [virtual]

Absolute get method.

Reads the value at the given index.

Parameters

index - the index in the Buffer where the short is to be read

Returns

the short that is located at the given index

Exceptions

IndexOutOfBoundsException - If index is not smaller than the buffer's limit

Implements `decaf::nio::ShortBuffer` (p. 2751).

6.621.2.8 virtual bool decaf::internal::nio::ShortArrayBuffer::hasArray () const
[inline, virtual]

Tells whether or not this buffer is backed by an accessible short array.

If this method returns true then the array and arrayOffset methods may safely be invoked. Sub-classes should override this method if they do not have a backing array as this class always returns true.

Returns

true if, and only if, this buffer is backed by an array and is not read-only

Implements **decaf::nio::ShortBuffer** (p. 2752).

6.621.2.9 virtual bool decaf::internal::nio::ShortArrayBuffer::isReadOnly () const
[inline, virtual]

Tells whether or not this buffer is read-only.

Returns

true if, and only if, this buffer is read-only

6.621.2.10 virtual ShortBuffer& decaf::internal::nio::ShortArrayBuffer::put
(std::size_t *index*, short *value*) throw
(lang::exceptions::IndexOutOfBoundsException,
decaf::nio::ReadOnlyBufferException) [virtual]

Writes the given doubles into this buffer at the given index.

Parameters

index - position in the Buffer to write the data

value - the doubles to write.

Returns

a reference to this buffer

Exceptions

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::ShortBuffer** (p. 2753).

6.621.2.11 virtual ShortBuffer& decaf::internal::nio::ShortArrayBuffer::put
(short *value*) throw (decaf::nio::BufferOverflowException,
decaf::nio::ReadOnlyBufferException) [virtual]

Writes the given doubles into this buffer at the current position, and then increments the position.

Parameters

value - the doubles value to be written

Returns

a reference to this buffer

Exceptions

BufferOverflowException - If this buffer's current position is not smaller than its limit

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::ShortBuffer** (p. 2753).

6.621.2.12 `virtual void decaf::internal::nio::ShortArrayBuffer::setReadOnly (bool value)` [inline, protected, virtual]

Sets this **ByteBuffer** (p. 806) as Read-Only.

Parameters

value - true if this buffer is to be read-only.

6.621.2.13 `virtual ShortBuffer* decaf::internal::nio::ShortArrayBuffer::slice ()`
`const` [virtual]

Creates a new **ShortBuffer** whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the newly create **ShortBuffer** which the caller owns.

Implements **decaf::nio::ShortBuffer** (p. 2755).

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/nio/ShortArrayBuffer.h`

6.622 decaf::nio::ShortBuffer Class Reference

This class defines four categories of operations upon short buffers:

```
#include <src/main/decaf/nio/ShortBuffer.h>
```

Inheritance diagram for **decaf::nio::ShortBuffer**:

Public Member Functions

- virtual **~ShortBuffer** ()
- virtual std::string **toString** () const
- virtual short * **array** ()=0 throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException)
Returns the short array that backs this buffer (optional operation).
- virtual std::size_t **arrayOffset** ()=0 throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException)
Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).
- virtual **ShortBuffer** * **asReadOnlyBuffer** () const =0
Creates a new, read-only short buffer that shares this buffer's content.
- virtual **ShortBuffer** & **compact** ()=0 throw (ReadOnlyBufferException)
Compacts this buffer.
- virtual **ShortBuffer** * **duplicate** ()=0
Creates a new short buffer that shares this buffer's content.
- virtual short **get** ()=0 throw (BufferUnderflowException)
Relative get method.
- virtual short **get** (std::size_t index) const =0 throw (lang::exceptions::IndexOutOfBoundsException)
Absolute get method.
- **ShortBuffer** & **get** (std::vector< short > buffer) throw (BufferUnderflowException)
Relative bulk get method.
- **ShortBuffer** & **get** (short *buffer, std::size_t offset, std::size_t length) throw (BufferUnderflowException, lang::exceptions::NullPointerException)
Relative bulk get method.
- virtual bool **hasArray** () const =0
Tells whether or not this buffer is backed by an accessible short array.
- **ShortBuffer** & **put** (**ShortBuffer** &src) throw (BufferOverflowException, ReadOnlyBufferException, lang::exceptions::IllegalArgumentException)
This method transfers the shorts remaining in the given source buffer into this buffer.
- **ShortBuffer** & **put** (const short *buffer, std::size_t offset, std::size_t length) throw (BufferOverflowException, ReadOnlyBufferException, lang::exceptions::NullPointerException)
This method transfers shorts into this buffer from the given source array.
- **ShortBuffer** & **put** (std::vector< short > &buffer) throw (BufferOverflowException, ReadOnlyBufferException)
This method transfers the entire content of the given source shorts array into this buffer.

- virtual **ShortBuffer** & **put** (short value)=0 throw (**BufferOverflowException**, **ReadOnlyBufferException**)
Writes the given shorts into this buffer at the current position, and then increments the position.
- virtual **ShortBuffer** & **put** (std::size_t index, short value)=0 throw (**lang::exceptions::IndexOutOfBoundsException**, **ReadOnlyBufferException**)
Writes the given shorts into this buffer at the given index.
- virtual **ShortBuffer** * **slice** () const =0
*Creates a new **ShortBuffer** (p. 2745) whose content is a shared subsequence of this buffer's content.*
- virtual int **compareTo** (const **ShortBuffer** &value) const
Compares this object with the specified object for order.
- virtual bool **equals** (const **ShortBuffer** &value) const
- virtual bool **operator==** (const **ShortBuffer** &value) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const **ShortBuffer** &value) const
Compares this object to another and returns true if this object is considered to be less than the one passed.

Static Public Member Functions

- static **ShortBuffer** * **allocate** (std::size_t capacity)
*Allocates a new **Double** buffer.*
- static **ShortBuffer** * **wrap** (short *array, std::size_t offset, std::size_t length) throw (**lang::exceptions::NullPointerException**)
*Wraps the passed buffer with a new **ShortBuffer** (p. 2745).*
- static **ShortBuffer** * **wrap** (std::vector< short > &buffer)
*Wraps the passed STL short Vector in a **ShortBuffer** (p. 2745).*

Protected Member Functions

- **ShortBuffer** (std::size_t capacity)
*Creates a **ShortBuffer** (p. 2745) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*

6.622.1 Detailed Description

This class defines four categories of operations upon short buffers: o Absolute and relative get and put methods that read and write single shorts; o Relative bulk get methods that transfer

contiguous sequences of shorts from this buffer into an array; and o Relative bulk put methods that transfer contiguous sequences of shorts from a short array or some other short buffer into this buffer o Methods for compacting, duplicating, and slicing a short buffer.

Double buffers can be created either by allocation, which allocates space for the buffer's content, by wrapping an existing short array into a buffer, or by creating a view of an existing byte buffer

Methods in this class that do not otherwise have a value to return are specified to return the buffer upon which they are invoked. This allows method invocations to be chained.

6.622.2 Constructor & Destructor Documentation

6.622.2.1 decaf::nio::ShortBuffer::ShortBuffer (std::size_t *capacity*) [protected]

Creates a **ShortBuffer** (p. 2745) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.

The array is initially created with all elements initialized to zero.

Parameters

capacity - size and limit of the **Buffer** (p. 741) in doubles

6.622.2.2 virtual decaf::nio::ShortBuffer::~~ShortBuffer () [inline, virtual]

6.622.3 Member Function Documentation

6.622.3.1 static ShortBuffer* decaf::nio::ShortBuffer::allocate (std::size_t *capacity*) [static]

Allocates a new Double buffer.

The new buffer's position will be zero, its limit will be its capacity, and its mark will be undefined. It will have a backing array, and its array offset will be zero.

Parameters

capacity - The size of the Double buffer in shorts

Returns

the **ShortBuffer** (p. 2745) that was allocated, caller owns.

6.622.3.2 virtual short* decaf::nio::ShortBuffer::array () throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException) [pure virtual]

Returns the short array that backs this buffer (optional operation).

Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

the array that backs this **Buffer** (p. 741)

Exceptions

ReadOnlyBufferException (p. 2520) if this **Buffer** (p. 741) is read only.

UnsupportedOperationException if the underlying store has no array.

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 2741).

6.622.3.3 `virtual std::size_t decaf::nio::ShortBuffer::arrayOffset ()
throw (decaf::lang::exceptions::UnsupportedOperationException,
ReadOnlyBufferException) [pure virtual]`

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

Invoke the `hasArray` method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns

The offset into the backing array where index zero starts.

Exceptions

ReadOnlyBufferException (p. 2520) if this **Buffer** (p. 741) is read only.

UnsupportedOperationException if the underlying store has no array.

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 2741).

6.622.3.4 `virtual ShortBuffer* decaf::nio::ShortBuffer::asReadOnlyBuffer ()
const [pure virtual]`

Creates a new, read-only short buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the `duplicate` method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns

The new, read-only short buffer which the caller then owns.

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 2742).

6.622.3.5 virtual ShortBuffer& decaf::nio::ShortBuffer::compact () throw (ReadOnlyBufferException) [pure virtual]

Compacts this buffer.

The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index $p = \text{position}()$ (p. 746) is copied to index zero, the byte at index $p + 1$ is copied to index one, and so forth until the byte at index $\text{limit}()$ (p. 745) - 1 is copied to index $n = \text{limit}()$ (p. 745) - 1 - p . The buffer's position is then set to $n+1$ and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns

a reference to this **ShortBuffer** (p. 2745)

Exceptions

ReadOnlyBufferException (p. 2520) - If this buffer is read-only

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 2742).

6.622.3.6 virtual int decaf::nio::ShortBuffer::compareTo (const ShortBuffer & value) const [virtual]

Compares this object with the specified object for order.

Returns a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

Parameters

value - the Object to be compared.

Returns

a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

6.622.3.7 virtual ShortBuffer* decaf::nio::ShortBuffer::duplicate () [pure virtual]

Creates a new short buffer that shares this buffer's content.

The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

a new short **Buffer** (p. 741) which the caller owns.

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 2743).

6.622.3.8 `virtual bool decaf::nio::ShortBuffer::equals (const ShortBuffer & value) const [virtual]`

Returns

true if this value is considered equal to the passed value.

6.622.3.9 `ShortBuffer& decaf::nio::ShortBuffer::get (std::vector< short > buffer) throw (BufferUnderflowException)`

Relative bulk get method.

This method transfers values from this buffer into the given destination vector. An invocation of this method of the form `src.get(a)` behaves in exactly the same way as the invocation. The vector must be sized to the amount of data that is to be read, that is to say, the caller should call `buffer.resize(N)` before calling this get method.

Returns

a reference to this **Buffer** (p. 741)

Exceptions

BufferUnderflowException (p. 774) - If there are fewer than length shorts remaining in this buffer

6.622.3.10 `virtual short decaf::nio::ShortBuffer::get (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException) [pure virtual]`

Absolute get method.

Reads the value at the given index.

Parameters

index - the index in the **Buffer** (p. 741) where the short is to be read

Returns

the short that is located at the given index

Exceptions

IndexOutOfBoundsException - If index is not smaller than the buffer's limit

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 2743).

6.622.3.11 `ShortBuffer& decaf::nio::ShortBuffer::get (short * buffer, std::size_t offset, std::size_t length) throw (BufferUnderflowException, lang::exceptions::NullPointerException)`

Relative bulk get method.

This method transfers shorts from this buffer into the given destination array. If there are fewer shorts remaining in the buffer than are required to satisfy the request, that is, if `length > remaining()` (p. 746), then no bytes are transferred and a **BufferUnderflowException** (p. 774) is thrown.

Otherwise, this method copies `length` shorts from this buffer into the given array, starting at the current position of this buffer and at the given offset in the array. The position of this buffer is then incremented by `length`.

Parameters

buffer - pointer to an allocated buffer to fill
offset - position in the buffer to start filling
length - amount of data to put in the passed buffer

Returns

a reference to this **Buffer** (p. 741)

Exceptions

BufferUnderflowException (p. 774) - If there are fewer than `length` shorts remaining in this buffer
NullPointerException if the passed buffer is null.

6.622.3.12 virtual short decaf::nio::ShortBuffer::get () throw (BufferUnderflowException) [pure virtual]

Relative get method.

Reads the value at this buffer's current position, and then increments the position.

Returns

the short at the current position

Exceptions

BufferUnderflowException (p. 774) if there no more data to return

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 2743).

6.622.3.13 virtual bool decaf::nio::ShortBuffer::hasArray () const [pure virtual]

Tells whether or not this buffer is backed by an accessible short array.

If this method returns true then the `array` and `arrayOffset` methods may safely be invoked. Sub-classes should override this method if they do not have a backing array as this class always returns true.

Returns

true if, and only if, this buffer is backed by an array and is not read-only

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 2744).

6.622.3.14 `virtual bool decaf::nio::ShortBuffer::operator< (const ShortBuffer & value) const [virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

Parameters

value - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

6.622.3.15 `virtual bool decaf::nio::ShortBuffer::operator== (const ShortBuffer & value) const [virtual]`

Compares equality between this object and the one passed.

Parameters

value - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

6.622.3.16 `virtual ShortBuffer& decaf::nio::ShortBuffer::put (short value) throw (BufferOverflowException, ReadOnlyBufferException) [pure virtual]`

Writes the given shorts into this buffer at the current position, and then increments the position.

Parameters

value - the shorts value to be written

Returns

a reference to this buffer

Exceptions

BufferOverflowException (p. 771) - If this buffer's current position is not smaller than its limit

ReadOnlyBufferException (p. 2520) - If this buffer is read-only

Implemented in `decaf::internal::nio::ShortArrayBuffer` (p. 2744).

6.622.3.17 `virtual ShortBuffer& decaf::nio::ShortBuffer::put (std::size_t index, short value) throw (lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException) [pure virtual]`

Writes the given shorts into this buffer at the given index.

Parameters

index - position in the **Buffer** (p. 741) to write the data
value - the shorts to write.

Returns

a reference to this buffer

Exceptions

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException (p. 2520) - If this buffer is read-only

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 2744).

6.622.3.18 **ShortBuffer& decaf::nio::ShortBuffer::put (const short
 * buffer, std::size_t offset, std::size_t length) throw
 (BufferOverflowException, ReadOnlyBufferException,
 lang::exceptions::NullPointerException)**

This method transfers shorts into this buffer from the given source array.

If there are more shorts to be copied from the array than remain in this buffer, that is, if `length > remaining()` (p. 746), then no shorts are transferred and a **BufferOverflowException** (p. 771) is thrown.

Otherwise, this method copies length bytes from the given array into this buffer, starting at the given offset in the array and at the current position of this buffer. The position of this buffer is then incremented by length.

Parameters

buffer- The array from which shorts are to be read
offset- The offset within the array of the first short to be read;
length - The number of shorts to be read from the given array

Returns

a reference to this buffer

Exceptions

BufferOverflowException (p. 771) - If there is insufficient space in this buffer

ReadOnlyBufferException (p. 2520) - If this buffer is read-only

NullPointerException if the passed buffer is null.

6.622.3.19 **ShortBuffer& decaf::nio::ShortBuffer::put (std::vector< short > &
 buffer) throw (BufferOverflowException, ReadOnlyBufferException)**

This method transfers the entire content of the given source shorts array into this buffer.

This is the same as calling `put(&buffer[0], 0, buffer.size()`

Parameters

buffer - The buffer whose contents are copied to this **ShortBuffer** (p. 2745)

Returns

a reference to this buffer

Exceptions

BufferOverflowException (p. 771) - If there is insufficient space in this buffer

ReadOnlyBufferException (p. 2520) - If this buffer is read-only

6.622.3.20 **ShortBuffer& decaf::nio::ShortBuffer::put (ShortBuffer & src) throw (BufferOverflowException, ReadOnlyBufferException, lang::exceptions::IllegalArgumentException)**

This method transfers the shorts remaining in the given source buffer into this buffer.

If there are more shorts remaining in the source buffer than in this buffer, that is, if `src.remaining() > remaining()` (p. 746), then no shorts are transferred and a **BufferOverflowException** (p. 771) is thrown.

Otherwise, this method copies `n = src.remaining()` shorts from the given buffer into this buffer, starting at each buffer's current position. The positions of both buffers are then incremented by `n`.

Parameters

src - the buffer to take shorts from an place in this one.

Returns

a reference to this buffer

Exceptions

BufferOverflowException (p. 771) - If there is insufficient space in this buffer for the remaining shorts in the source buffer

IllegalArgumentException - If the source buffer is this buffer

ReadOnlyBufferException (p. 2520) - If this buffer is read-only

6.622.3.21 **virtual ShortBuffer* decaf::nio::ShortBuffer::slice () const [pure virtual]**

Creates a new **ShortBuffer** (p.2745) whose content is a shared subsequence of this buffer's content.

The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns

the newly create **ShortBuffer** (p.2745) which the caller owns.

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p.2745).

6.622.3.22 virtual std::string decaf::nio::ShortBuffer::toString () const [virtual]**Returns**

a std::string describing this object

6.622.3.23 static ShortBuffer* decaf::nio::ShortBuffer::wrap (short * array, std::size_t offset, std::size_t length) throw (lang::exceptions::NullPointerException) [static]

Wraps the passed buffer with a new **ShortBuffer** (p.2745).

The new buffer will be backed by the given short array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be array.length, its position will be offset, its limit will be offset + length, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

array - The array that will back the new buffer

offset - The offset of the subarray to be used

length - The length of the subarray to be used

Returns

a new **ShortBuffer** (p.2745) that is backed by buffer, caller owns.

6.622.3.24 static ShortBuffer* decaf::nio::ShortBuffer::wrap (std::vector< short > & buffer) [static]

Wraps the passed STL short Vector in a **ShortBuffer** (p.2745).

The new buffer will be backed by the given short array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be buffer.size(), its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters

buffer - The vector that will back the new buffer, the vector must have been sized to the desired size already by calling vector.resize(N).

Returns

a new **ShortBuffer** (p.2745) that is backed by buffer, caller owns.

The documentation for this class was generated from the following file:

- src/main/decaf/nio/**ShortBuffer.h**

6.623 activemq::commands::ShutdownInfo Class Reference

```
#include <src/main/activemq/commands/ShutdownInfo.h>
```

Inheritance diagram for activemq::commands::ShutdownInfo:

Public Member Functions

- **ShutdownInfo** ()
- virtual **~ShutdownInfo** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **ShutdownInfo * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1372) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent.*
- virtual bool **isShutdownInfo** () const
- virtual **Pointer< Command > visit** (activemq::state::CommandVisitor *visitor) throw (exceptions::ActiveMQException)
Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID _SHUTDOWNINFO** = 11

Protected Member Functions

- **ShutdownInfo** (const **ShutdownInfo** &)
- **ShutdownInfo** & **operator=** (const **ShutdownInfo** &)

6.623.1 Constructor & Destructor Documentation

6.623.1.1 `activemq::commands::ShutdownInfo::ShutdownInfo (const ShutdownInfo &)` [inline, protected]

6.623.1.2 `activemq::commands::ShutdownInfo::ShutdownInfo ()`

6.623.1.3 `virtual activemq::commands::ShutdownInfo::~~ShutdownInfo ()` [virtual]

6.623.2 Member Function Documentation

6.623.2.1 `virtual ShutdownInfo* activemq::commands::ShutdownInfo::cloneDataStructure () const` [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1373).

6.623.2.2 `virtual void activemq::commands::ShutdownInfo::copyDataStructure (const DataStructure * src)` [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

Reimplemented from `activemq::commands::BaseCommand` (p. 598).

6.623.2.3 `virtual bool activemq::commands::ShutdownInfo::equals (const DataStructure * value) const` [virtual]

Compares the `DataStructure` (p. 1372) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 598).

6.623.2.4 `virtual unsigned char activemq::commands::ShutdownInfo::getDataStructureType () const` [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataSet** (p. 1372) type copy.

Implements **activemq::commands::DataSet** (p. 1375).

6.623.2.5 `virtual bool activemq::commands::ShutdownInfo::isShutdownInfo ()
const [inline, virtual]`

Returns

an answer of true to the **isShutdownInfo()** (p. 2759) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 602).

6.623.2.6 `ShutdownInfo& activemq::commands::ShutdownInfo::operator= (const
ShutdownInfo &) [inline, protected]`

6.623.2.7 `virtual std::string activemq::commands::ShutdownInfo::toString ()
const [virtual]`

Returns a string containing the information for this **DataSet** (p. 1372) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 602).

6.623.2.8 `virtual Pointer<Command> activemq::commands::ShutdownInfo::visit
(activemq::state::CommandVisitor * visitor) throw (
exceptions::ActiveMQException) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

Returns

a **Response** (p. 2611) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 995).

6.623.3 Field Documentation

6.623.3.1 `const unsigned char activemq::commands::ShutdownInfo::ID_-
SHUTDOWNINFO = 11 [static]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/ShutdownInfo.h`

6.624 activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 2760).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/ShutdownInfoMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller:

Public Member Functions

- **ShutdownInfoMarshaller** ()
- virtual **~ShutdownInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.624.1 Detailed Description

Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 2760). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.624.2 Constructor & Destructor Documentation

6.624.2.1 `activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller::ShutdownInfoMarshaller()` [inline]

6.624.2.2 `virtual activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller::~~ShutdownInfoMarshaller()` [inline, virtual]

6.624.3 Member Function Documentation

6.624.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.624.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.624.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 611).

6.624.3.4 virtual void activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 612).

6.624.3.5 virtual int activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 613).

6.624.3.6 virtual void activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions

- IOException* if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 614).

```
6.624.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 615).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/ShutdownInfoMarshaller.h`

6.625 **activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller** Class Reference

Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 2763).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/ShutdownInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller**:

Public Member Functions

- **ShutdownInfoMarshaller** ()
- virtual **~ShutdownInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.625.1 Detailed Description

Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 2763). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.625.2 Constructor & Destructor Documentation

6.625.2.1 `activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller::ShutdownInfoMarshaller()` [inline]

6.625.2.2 `virtual activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller::~~ShutdownInfoMarshaller()` [inline, virtual]

6.625.3 Member Function Documentation

6.625.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.625.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.625.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 618).

6.625.3.4 virtual void activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 619).

6.625.3.5 virtual int activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 620).

6.625.3.6 virtual void activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 621).

```
6.625.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 622).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/ShutdownInfoMarshaller.h`

6.626 **activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller** Class Reference

Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 2767).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/ShutdownInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller**:

Public Member Functions

- **ShutdownInfoMarshaller** ()
- virtual **~ShutdownInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.626.1 Detailed Description

Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 2767). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.626.2 Constructor & Destructor Documentation

6.626.2.1 `activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller::ShutdownInfoMarshaller()` [inline]

6.626.2.2 `virtual activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller::~~ShutdownInfoMarshaller()` [inline, virtual]

6.626.3 Member Function Documentation

6.626.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.626.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.626.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 604).

6.626.3.4 virtual void activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 605).

6.626.3.5 virtual int activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 606).

6.626.3.6 virtual void activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 608).

```
6.626.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 609).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/ShutdownInfoMarshaller.h`

6.627 **activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller** Class Reference

Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 2771).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/ShutdownInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller**:

Public Member Functions

- **ShutdownInfoMarshaller** ()
- virtual **~ShutdownInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.627.1 Detailed Description

Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 2771). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.627.2 Constructor & Destructor Documentation

6.627.2.1 `activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller::ShutdownInfoMarshaller()` [inline]

6.627.2.2 `virtual activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller::~~ShutdownInfoMarshaller()` [inline, virtual]

6.627.3 Member Function Documentation

6.627.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.627.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.627.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 631).

6.627.3.4 virtual void activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 632).

6.627.3.5 virtual int activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 633).

6.627.3.6 virtual void activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 634).

```
6.627.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 635).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/ShutdownInfoMarshaller.h`

6.628 **activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller** Class Reference

Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 2775).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/ShutdownInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller**:

Public Member Functions

- **ShutdownInfoMarshaller** ()
- virtual **~ShutdownInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.628.1 Detailed Description

Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 2775). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.628.2 Constructor & Destructor Documentation

6.628.2.1 `activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller::ShutdownInfoMarshaller()` [inline]

6.628.2.2 `virtual activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller::~~ShutdownInfoMarshaller()` [inline, virtual]

6.628.3 Member Function Documentation

6.628.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.628.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.628.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 624).

6.628.3.4 virtual void activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 625).

6.628.3.5 virtual int activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 626).

6.628.3.6 virtual void activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 628).

```
6.628.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 629).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/ShutdownInfoMarshaller.h`

6.629 decaf::security::SignatureException Class Reference

```
#include <src/main/decaf/security/SignatureException.h>
```

Inheritance diagram for `decaf::security::SignatureException`:

Public Member Functions

- **SignatureException** () throw ()
Default Constructor.
- **SignatureException** (const Exception &ex) throw ()
Conversion Constructor from some other Exception.
- **SignatureException** (const **SignatureException** &ex) throw ()
Copy Constructor.
- **SignatureException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **SignatureException** (const std::exception *cause) throw ()
Constructor.
- **SignatureException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **SignatureException** * clone () const
Clones this exception.
- virtual ~**SignatureException** () throw ()

6.629.1 Constructor & Destructor Documentation

6.629.1.1 decaf::security::SignatureException::SignatureException () throw () [inline]

Default Constructor.

6.629.1.2 decaf::security::SignatureException::SignatureException (const Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

Parameters

ex An exception that should become this type of Exception

6.629.1.3 decaf::security::SignatureException::SignatureException (const SignatureException & ex) throw () [inline]

Copy Constructor.

Parameters

ex An exception that should become this type of Exception

6.629.1.4 `decaf::security::SignatureException::SignatureException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs
lineNumber The line number where the exception occurred.
cause The exception that was the cause for this one to be thrown.
msg The message to report
 ... list of primitives that are formatted into the message

6.629.1.5 `decaf::security::SignatureException::SignatureException (const std::exception * cause) throw () [inline]`

Constructor.

Parameters

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.629.1.6 `decaf::security::SignatureException::SignatureException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file name where exception occurs
lineNumber line number where the exception occurred.
msg message to report
 ... list of primitives that are formatted into the message

6.629.1.7 `virtual decaf::security::SignatureException::~SignatureException () throw () [inline, virtual]`

6.629.2 Member Function Documentation

6.629.2.1 `virtual SignatureException* decaf::security::SignatureException::clone () const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

A deep copy of this exception.

Reimplemented from **decaf::security::GeneralSecurityException** (p. 1604).

The documentation for this class was generated from the following file:

- `src/main/decaf/security/SignatureException.h`

6.630 decaf::util::logging::SimpleFormatter Class Reference

Print a brief summary of the **LogRecord** (p. 1928) in a human readable format.

```
#include <src/main/decaf/util/logging/SimpleFormatter.h>
```

Inheritance diagram for decaf::util::logging::SimpleFormatter:

Public Member Functions

- **SimpleFormatter** ()
- virtual **~SimpleFormatter** ()
- virtual std::string **format** (const **LogRecord** &record) const
Format the given log record and return the formatted string.
- virtual std::string **formatMessage** (const **LogRecord** &record) const
Format the message string from a log record.
- virtual std::string **getHead** (const **Handler** *handler)
Return the header string for a set of formatted records.
- virtual std::string **getTail** (const **Handler** *handler)
Return the tail string for a set of formatted records.

6.630.1 Detailed Description

Print a brief summary of the **LogRecord** (p. 1928) in a human readable format. The summary will typically be 1 or 2 lines.

6.630.2 Constructor & Destructor Documentation

6.630.2.1 `decaf::util::logging::SimpleFormatter::SimpleFormatter ()` [inline]

6.630.2.2 `virtual decaf::util::logging::SimpleFormatter::~~SimpleFormatter ()`
[inline, virtual]

6.630.3 Member Function Documentation

6.630.3.1 `virtual std::string decaf::util::logging::SimpleFormatter::format (const LogRecord & record) const` [inline, virtual]

Format the given log record and return the formatted string.

Parameters

record The Log Record to Format

Implements `decaf::util::logging::Formatter` (p. 1596).

6.630.3.2 `virtual std::string decaf::util::logging::SimpleFormatter::formatMessage (const LogRecord & record) const` [inline, virtual]

Format the message string from a log record.

Parameters

record The Log Record to Format

Implements `decaf::util::logging::Formatter` (p. 1596).

References `decaf::util::logging::LogRecord::getMessage()`.

6.630.3.3 `virtual std::string decaf::util::logging::SimpleFormatter::getHead (const Handler * handler)` [inline, virtual]

Return the header string for a set of formatted records.

In the default implementation this method should return empty string

Parameters

handler the target handler, can be null

Returns

empty string

Implements `decaf::util::logging::Formatter` (p. 1597).

6.630.3.4 `virtual std::string decaf::util::logging::SimpleFormatter::getTail (const Handler * handler)` [inline, virtual]

Return the tail string for a set of formatted records.

In the default implementation this method should return empty string

Parameters

handler the target handler, can be null

Returns

empty string

Implements **decaf::util::logging::Formatter** (p. 1597).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/logging/SimpleFormatter.h`

6.631 decaf::util::logging::SimpleLogger Class Reference

```
#include <src/main/decaf/util/logging/SimpleLogger.h>
```

Public Member Functions

- **SimpleLogger** (const std::string &name)
Constructor.
- virtual **~SimpleLogger** ()
Destructor.
- virtual void **mark** (const std::string &message)
Log a Mark Block Level Log.
- virtual void **debug** (const std::string &file, const int line, const std::string &message)
Log a Debug Level Log.
- virtual void **info** (const std::string &file, const int line, const std::string &message)
Log a Informational Level Log.
- virtual void **warn** (const std::string &file, const int line, const std::string &message)
Log a Warning Level Log.
- virtual void **error** (const std::string &file, const int line, const std::string &message)
Log a Error Level Log.
- virtual void **fatal** (const std::string &file, const int line, const std::string &message)
Log a Fatal Level Log.
- virtual void **log** (const std::string &message)
No-frills log.

6.631.1 Constructor & Destructor Documentation

6.631.1.1 `decaf::util::logging::SimpleLogger::SimpleLogger (const std::string & name)`

Constructor.

6.631.1.2 `virtual decaf::util::logging::SimpleLogger::~SimpleLogger () [virtual]`

Destructor.

6.631.2 Member Function Documentation

6.631.2.1 `virtual void decaf::util::logging::SimpleLogger::debug (const std::string & file, const int line, const std::string & message) [virtual]`

Log a Debug Level Log.

6.631.2.2 `virtual void decaf::util::logging::SimpleLogger::error (const std::string & file, const int line, const std::string & message) [virtual]`

Log a Error Level Log.

6.631.2.3 `virtual void decaf::util::logging::SimpleLogger::fatal (const std::string & file, const int line, const std::string & message) [virtual]`

Log a Fatal Level Log.

6.631.2.4 `virtual void decaf::util::logging::SimpleLogger::info (const std::string & file, const int line, const std::string & message) [virtual]`

Log a Informational Level Log.

6.631.2.5 `virtual void decaf::util::logging::SimpleLogger::log (const std::string & message) [virtual]`

No-frills log.

6.631.2.6 `virtual void decaf::util::logging::SimpleLogger::mark (const std::string & message) [virtual]`

Log a Mark Block Level Log.

6.631.2.7 `virtual void decaf::util::logging::SimpleLogger::warn (const std::string & file, const int line, const std::string & message) [virtual]`

Log a Warning Level Log.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/logging/SimpleLogger.h`

6.632 decaf::net::Socket Class Reference

```
#include <src/main/decaf/net/Socket.h>
```

Inheritance diagram for `decaf::net::Socket`:

Public Types

- `typedef apr_socket_t * SocketHandle`
Define the SocketHandle type.
- `typedef apr_sockaddr_t * SocketAddress`
Define the SocketAddress type.

Public Member Functions

- `virtual ~Socket ()`
- `virtual void connect (const char *host, int port)=0 throw (SocketException)`
Connects to the specified destination.
- `virtual bool isConnected () const =0`
Indicates whether or not this socket is connected to a destination.
- `virtual io::InputStream * getInputStream ()=0`
Gets the InputStream for this socket.
- `virtual io::OutputStream * getOutputStream ()=0`
Gets the OutputStream for this socket.
- `virtual int getSoLinger () const =0 throw (SocketException)`
Gets the linger time.
- `virtual void setSoLinger (int linger)=0 throw (SocketException)`
Sets the linger time.
- `virtual bool getKeepAlive () const =0 throw (SocketException)`
Gets the keep alive flag.
- `virtual void setKeepAlive (bool keepAlive)=0 throw (SocketException)`
Enables/disables the keep alive flag.
- `virtual int getReceiveBufferSize () const =0 throw (SocketException)`
Gets the receive buffer size.

- virtual void **setReceiveBufferSize** (int size)=0 throw (SocketException)
Sets the receive buffer size.
- virtual bool **getReuseAddress** () const =0 throw (SocketException)
Gets the reuse address flag.
- virtual void **setReuseAddress** (bool reuse)=0 throw (SocketException)
Sets the reuse address flag.
- virtual int **getSendBufferSize** () const =0 throw (SocketException)
Gets the send buffer size.
- virtual void **setSendBufferSize** (int size)=0 throw (SocketException)
Sets the send buffer size.
- virtual int **getSoTimeout** () const =0 throw (SocketException)
Gets the timeout for socket operations.
- virtual void **setSoTimeout** (int timeout)=0 throw (SocketException)
Sets the timeout for socket operations.

6.632.1 Member Typedef Documentation

6.632.1.1 typedef apr_sockaddr_t* decaf::net::Socket::SocketAddress

Define the SocketAddress type.

6.632.1.2 typedef apr_socket_t* decaf::net::Socket::SocketHandle

Define the SocketHandle type.

6.632.2 Constructor & Destructor Documentation

6.632.2.1 virtual decaf::net::Socket::~~Socket () [inline, virtual]

6.632.3 Member Function Documentation

6.632.3.1 virtual void decaf::net::Socket::connect (const char * *host*, int *port*) throw (SocketException) [pure virtual]

Connects to the specified destination.

Closes this socket if connected to another destination.

Parameters

host The host of the server to connect to.

port The port of the server to connect to.

Exceptions

IOException Thrown if a failure occurred in the connect.

Implemented in **decaf::net::BufferedSocket** (p. 757), and **decaf::net::TcpSocket** (p. 2962).

6.632.3.2 `virtual io::InputStream* decaf::net::Socket::getInputStream () [pure virtual]`

Gets the InputStream for this socket.

Returns

The InputStream for this socket. NULL if not connected.

Implemented in **decaf::net::BufferedSocket** (p. 757), and **decaf::net::TcpSocket** (p. 2962).

6.632.3.3 `virtual bool decaf::net::Socket::getKeepAlive () const throw (SocketException) [pure virtual]`

Gets the keep alive flag.

Returns

True if keep alive is enabled.

Exceptions

SocketException (p. 2793) if the operation fails.

Implemented in **decaf::net::BufferedSocket** (p. 757), and **decaf::net::TcpSocket** (p. 2962).

Referenced by **decaf::net::BufferedSocket::getKeepAlive()**.

6.632.3.4 `virtual io::OutputStream* decaf::net::Socket::getOutputStream () [pure virtual]`

Gets the OutputStream for this socket.

Returns

the OutputStream for this socket. NULL if not connected.

Implemented in **decaf::net::BufferedSocket** (p. 758), and **decaf::net::TcpSocket** (p. 2963).

6.632.3.5 `virtual int decaf::net::Socket::getReceiveBufferSize () const throw (SocketException) [pure virtual]`

Gets the receive buffer size.

Returns

the receive buffer size in bytes.

Exceptions

SocketException (p. 2793) if the operation fails.

Implemented in **decaf::net::BufferedSocket** (p. 758), and **decaf::net::TcpSocket** (p. 2963).

Referenced by **decaf::net::BufferedSocket::getReceiveBufferSize()**.

6.632.3.6 `virtual bool decaf::net::Socket::getReuseAddress () const throw (SocketException) [pure virtual]`

Gets the reuse address flag.

Returns

True if the address can be reused.

Exceptions

SocketException (p. 2793) if the operation fails.

Implemented in **decaf::net::BufferedSocket** (p. 758), and **decaf::net::TcpSocket** (p. 2963).

Referenced by **decaf::net::BufferedSocket::getReuseAddress()**.

6.632.3.7 `virtual int decaf::net::Socket::getSendBufferSize () const throw (SocketException) [pure virtual]`

Gets the send buffer size.

Returns

the size in bytes of the send buffer.

Exceptions

SocketException (p. 2793) if the operation fails.

Implemented in **decaf::net::BufferedSocket** (p. 759), and **decaf::net::TcpSocket** (p. 2963).

Referenced by **decaf::net::BufferedSocket::getSendBufferSize()**.

6.632.3.8 `virtual int decaf::net::Socket::getSoLinger () const throw (SocketException) [pure virtual]`

Gets the linger time.

Returns

The linger time in seconds.

Exceptions

SocketException (p. 2793) if the operation fails.

Implemented in **decaf::net::BufferedSocket** (p. 759), and **decaf::net::TcpSocket** (p. 2964).

Referenced by **decaf::net::BufferedSocket::getSoLinger()**.

6.632.3.9 virtual int decaf::net::Socket::getSoTimeout () const throw (SocketException) [pure virtual]

Gets the timeout for socket operations.

Returns

The timeout in milliseconds for socket operations.

Exceptions

SocketException (p. 2793) Thrown if unable to retrieve the information.

Implemented in **decaf::net::BufferedSocket** (p. 759), and **decaf::net::TcpSocket** (p. 2964).

Referenced by `decaf::net::BufferedSocket::getSoTimeout()`.

6.632.3.10 virtual bool decaf::net::Socket::isConnected () const [pure virtual]

Indicates whether or not this socket is connected to a destination.

Returns

true if connected

Implemented in **decaf::net::BufferedSocket** (p. 759), and **decaf::net::TcpSocket** (p. 2965).

Referenced by `decaf::net::BufferedSocket::isConnected()`.

6.632.3.11 virtual void decaf::net::Socket::setKeepAlive (bool *keepAlive*) throw (SocketException) [pure virtual]

Enables/disables the keep alive flag.

Parameters

keepAlive If true, enables the flag.

Exceptions

SocketException (p. 2793) if the operation fails.

Implemented in **decaf::net::BufferedSocket** (p. 760), and **decaf::net::TcpSocket** (p. 2965).

Referenced by `decaf::net::BufferedSocket::setKeepAlive()`.

6.632.3.12 virtual void decaf::net::Socket::setReceiveBufferSize (int *size*) throw (SocketException) [pure virtual]

Sets the receive buffer size.

Parameters

size Number of bytes to set the receive buffer to.

Exceptions

SocketException (p. 2793) if the operation fails.

Implemented in **decaf::net::BufferedSocket** (p. 760), and **decaf::net::TcpSocket** (p. 2965).

Referenced by **decaf::net::BufferedSocket::setReceiveBufferSize()**.

6.632.3.13 **virtual void decaf::net::Socket::setReuseAddress (bool *reuse*) throw (SocketException)** [pure virtual]

Sets the reuse address flag.

Parameters

reuse If true, sets the flag.

Exceptions

SocketException (p. 2793) if the operation fails.

Implemented in **decaf::net::BufferedSocket** (p. 760), and **decaf::net::TcpSocket** (p. 2965).

Referenced by **decaf::net::BufferedSocket::setReuseAddress()**.

6.632.3.14 **virtual void decaf::net::Socket::setSendBufferSize (int *size*) throw (SocketException)** [pure virtual]

Sets the send buffer size.

Parameters

size The number of bytes to set the send buffer to.

Exceptions

SocketException (p. 2793) if the operation fails.

Implemented in **decaf::net::BufferedSocket** (p. 761), and **decaf::net::TcpSocket** (p. 2966).

Referenced by **decaf::net::BufferedSocket::setSendBufferSize()**.

6.632.3.15 **virtual void decaf::net::Socket::setSoLinger (int *linger*) throw (SocketException)** [pure virtual]

Sets the linger time.

Parameters

linger The linger time in seconds. If 0, linger is off.

Exceptions

SocketException (p. 2793) if the operation fails.

Implemented in **decaf::net::BufferedSocket** (p. 761), and **decaf::net::TcpSocket** (p. 2966).

Referenced by **decaf::net::BufferedSocket::setSoLinger()**.

6.632.3.16 `virtual void decaf::net::Socket::setSoTimeout (int timeout) throw (SocketException)` [pure virtual]

Sets the timeout for socket operations.

Parameters

timeout The timeout in milliseconds for socket operations.

Exceptions

SocketException (p. 2793) Thrown if unable to set the information.

Implemented in `decaf::net::BufferedSocket` (p. 761), and `decaf::net::TcpSocket` (p. 2966).

Referenced by `decaf::net::BufferedSocket::setSoTimeout()`.

The documentation for this class was generated from the following file:

- `src/main/decaf/net/Socket.h`

6.633 decaf::net::SocketError Class Reference

Static utility class to simplify handling of error codes for socket operations.

`#include <src/main/decaf/net/SocketError.h>`

Static Public Member Functions

- static int `getErrorCode ()`
Gets the last error appropriate for the platform.
- static std::string `getErrorString ()`
Gets the string description for the last error.

6.633.1 Detailed Description

Static utility class to simplify handling of error codes for socket operations.

6.633.2 Member Function Documentation

6.633.2.1 `static int decaf::net::SocketError::getErrorCode ()` [static]

Gets the last error appropriate for the platform.

6.633.2.2 `static std::string decaf::net::SocketError::getErrorString ()` [static]

Gets the string description for the last error.

The documentation for this class was generated from the following file:

- `src/main/decaf/net/SocketError.h`

6.634 decaf::net::SocketException Class Reference

Exception for errors when manipulating sockets.

```
#include <src/main/decaf/net/SocketException.h>
```

Inheritance diagram for decaf::net::SocketException:

Public Member Functions

- **SocketException** () throw ()
- **SocketException** (const **lang::Exception** &ex) throw ()
- **SocketException** (const **SocketException** &ex) throw ()
- **SocketException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **SocketException** (const std::exception *cause) throw ()
Constructor.
- **SocketException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **SocketException** * **clone** () const
Clones this exception.
- virtual ~**SocketException** () throw ()

6.634.1 Detailed Description

Exception for errors when manipulating sockets.

6.634.2 Constructor & Destructor Documentation

6.634.2.1 decaf::net::SocketException::SocketException () throw () [inline]

6.634.2.2 decaf::net::SocketException::SocketException (const lang::Exception &ex) throw () [inline]

6.634.2.3 decaf::net::SocketException::SocketException (const SocketException &ex) throw () [inline]

6.634.2.4 decaf::net::SocketException::SocketException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs
lineNumber The line number where the exception occurred.
cause The exception that was the cause for this one to be thrown.
msg The message to report
... list of primitives that are formatted into the message

6.634.2.5 decaf::net::SocketException::SocketException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.634.2.6 decaf::net::SocketException::SocketException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs
lineNumber The line number where the exception occurred.
msg The message to report
... list of primitives that are formatted into the message

6.634.2.7 virtual decaf::net::SocketException::~SocketException () throw () [inline, virtual]

6.634.3 Member Function Documentation

6.634.3.1 virtual SocketException* decaf::net::SocketException::clone () const [inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::io::IOException** (p. 1709).

Reimplemented in **decaf::net::BindException** (p. 665), **decaf::net::ConnectException** (p. 1051), **decaf::net::NoRouteToHostException** (p. 2271), and **decaf::net::PortUnreachableException** (p. 2370).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/SocketException.h`

6.635 decaf::net::SocketFactory Class Reference

Socket (p. 2786) Factory implementation for use in Creating Sockets.

```
#include <src/main/decaf/net/SocketFactory.h>
```

Public Member Functions

- virtual `~SocketFactory()`

Static Public Member Functions

- static **Socket** * **createSocket** (const std::string &uri, const **util::Properties** &properties) throw (**SocketException**)

*Creates and returns a **Socket** (p. 2786) derived Object based on the values defined in the Properties Object that is passed in.*

6.635.1 Detailed Description

Socket (p. 2786) Factory implementation for use in Creating Sockets. Property Options:

Name Value

inputBufferSize size in bytes of the buffered input stream buffer. Defaults to 10000.

outputBufferSize size in bytes of the buffered output stream buffer. Defaults to 10000.

soLinger linger time for the socket (in microseconds). Defaults to 0.

soKeepAlive keep alive flag for the socket (true/false). Defaults to false.

soReceiveBufferSize The size of the socket receive buffer (in bytes). Defaults to 2MB.

soSendBufferSize The size of the socket send buffer (in bytes). Defaults to 2MB.

soTimeout The timeout of socket IO operations (in microseconds). Defaults to 10000

See also

Socket (p. 2786)

6.635.2 Constructor & Destructor Documentation

6.635.2.1 `virtual decaf::net::SocketFactory::~SocketFactory () [inline, virtual]`

6.635.3 Member Function Documentation

6.635.3.1 `static Socket* decaf::net::SocketFactory::createSocket (const std::string & uri, const util::Properties & properties) throw (SocketException) [static]`

Creates and returns a **Socket** (p. 2786) derived Object based on the values defined in the Properties Object that is passed in.

Parameters

- uri* the **URI** (p. 3096) for the **Socket** (p. 2786) Connection.
- properties* A Properties object that contains configuration details.

Exceptions

SocketException.

The documentation for this class was generated from the following file:

- `src/main/decaf/net/SocketFactory.h`

6.636 decaf::net::SocketInputStream Class Reference

Input stream for performing reads on a socket.

```
#include <src/main/decaf/net/SocketInputStream.h>
```

Inheritance diagram for decaf::net::SocketInputStream:

Public Member Functions

- **SocketInputStream** (**Socket::SocketHandle** socket)
Constructor.
- virtual **~SocketInputStream** ()
Destructor.
- virtual `std::size_t available () const throw (io::IOException)`
Returns the number of bytes available on the socket to be read right now.
- virtual `int read () throw (io::IOException)`
Reads a single byte from the buffer.

- virtual int **read** (unsigned char *buffer, std::size_t offset, std::size_t bufferSize) throw (io::IOException, lang::exceptions::NullPointerException)
Reads an array of bytes from the buffer.
- virtual void **close** () throw (decaf::io::IOException)
Close - does nothing.
- virtual std::size_t **skip** (std::size_t num) throw (io::IOException, lang::exceptions::UnsupportedOperationException)
Not supported.
- virtual void **mark** (int readLimit DECAF_UNUSED)
Marks the current position in the stream A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.
- virtual void **reset** () throw (io::IOException)
Repositions this stream to the position at the time the mark method was last called on this input stream.
- virtual bool **markSupported** () const
Determines if this input stream supports the mark and reset methods.
- virtual void **lock** () throw (decaf::lang::exceptions::RuntimeException)
Locks the object.
- virtual bool **tryLock** () throw (decaf::lang::exceptions::RuntimeException)
Attempts to Lock the object, if the lock is already held by another thread than this method returns false.
- virtual void **unlock** () throw (decaf::lang::exceptions::RuntimeException)
Unlocks the object.
- virtual void **wait** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (long long millisecs) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (long long millisecs, int nanos) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **notify** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)
Signals a waiter on this object that it can now wake up and continue.

- virtual void **notifyAll** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)

Signals the waiters on this object that it can now wake up and continue.

6.636.1 Detailed Description

Input stream for performing reads on a socket. This class will only work properly for blocking sockets.

6.636.2 Constructor & Destructor Documentation

6.636.2.1 decaf::net::SocketInputStream::SocketInputStream (Socket::SocketHandle *socket*)

Constructor.

Parameters

socket the socket handle.

6.636.2.2 virtual decaf::net::SocketInputStream::~~SocketInputStream () [virtual]

Destructor.

6.636.3 Member Function Documentation

6.636.3.1 virtual std::size_t decaf::net::SocketInputStream::available () const throw (io::IOException) [virtual]

Returns the number of bytes available on the socket to be read right now.

Returns

The number of bytes currently available to be read on the socket.

Implements **decaf::io::InputStream** (p. 1631).

6.636.3.2 virtual void decaf::net::SocketInputStream::close () throw (decaf::io::IOException) [virtual]

Close - does nothing.

It is the responsibility of the owner of the socket object to close it.

Exceptions

IOException

6.636.3.3 `virtual void decaf::net::SocketInputStream::lock () throw (decaf::lang::exceptions::RuntimeException)` [inline, virtual]

Locks the object.

Exceptions

RuntimeException if an error occurs while locking the object.

Implements `decaf::util::concurrent::Synchronizable` (p. 2932).

6.636.3.4 `virtual void decaf::net::SocketInputStream::mark (int readLimit DECAF_UNUSED)` [inline, virtual]

Marks the current position in the stream. A subsequent call to the `reset` method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.

If a stream instance reports that marks are supported then the stream will ensure that the same bytes can be read again after the `reset` method is called so long the `readLimit` is not reached.

Parameters

readLimit - max bytes read before marked position is invalid.

Implements `decaf::io::InputStream` (p. 1631).

6.636.3.5 `virtual bool decaf::net::SocketInputStream::markSupported () const` [inline, virtual]

Determines if this input stream supports the `mark` and `reset` methods.

Whether or not `mark` and `reset` are supported is an invariant property of a particular input stream instance.

Returns

true if this stream instance supports marks

Implements `decaf::io::InputStream` (p. 1631).

6.636.3.6 `virtual void decaf::net::SocketInputStream::notify () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)` [inline, virtual]

Signals a waiter on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

RuntimeException if an error occurs while notifying one of the waiting threads.

Implements `decaf::util::concurrent::Synchronizable` (p. 2933).

6.636.3.7 `virtual void decaf::net::SocketInputStream::notifyAll () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException) [inline, virtual]`

Signals the waiters on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

RuntimeException if an error occurs while notifying the waiting threads.

Implements `decaf::util::concurrent::Synchronizable` (p. 2934).

6.636.3.8 `virtual int decaf::net::SocketInputStream::read () throw (io::IOException) [virtual]`

Reads a single byte from the buffer.

If no data is available, blocks until their is.

Returns

The next byte.

Exceptions

IOException thrown if an error occurs.

Implements `decaf::io::InputStream` (p. 1632).

6.636.3.9 `virtual int decaf::net::SocketInputStream::read (unsigned char * buffer, std::size_t offset, std::size_t bufferSize) throw (io::IOException, lang::exceptions::NullPointerException) [virtual]`

Reads an array of bytes from the buffer.

If the desired amount of data is not currently available, this operation will block until the appropriate amount of data is available.

Parameters

buffer (out) the target buffer

offset the position in the buffer to start from.

bufferSize the size of the output buffer.

Returns

the number of bytes read. or -1 if EOF

Exceptions

IOException f an error occurs.

Implements `decaf::io::InputStream` (p. 1632).

6.636.3.10 `virtual void decaf::net::SocketInputStream::reset () throw (io::IOException)` [inline, virtual]

Repositions this stream to the position at the time the mark method was last called on this input stream.

If the method markSupported returns true, then: * If the method mark has not been called since the stream was created, or the number of bytes read from the stream since mark was last called is larger than the argument to mark at that last call, then an IOException might be thrown. * If such an IOException is not thrown, then the stream is reset to a state such that all the bytes read since the most recent call to mark (or since the start of the file, if mark has not been called) will be resupplied to subsequent callers of the read method, followed by any bytes that otherwise would have been the next input data as of the time of the call to reset. If the method markSupported returns false, then: * The call to reset may throw an IOException. * If an IOException is not thrown, then the stream is reset to a fixed state that depends on the particular type of the input stream and how it was created. The bytes that will be supplied to subsequent callers of the read method depend on the particular type of the input stream.

Exceptions

IOException

Implements `decaf::io::InputStream` (p. 1633).

6.636.3.11 `virtual std::size_t decaf::net::SocketInputStream::skip (std::size_t num) throw (io::IOException, lang::exceptions::UnsupportedOperationException)` [virtual]

Not supported.

Exceptions

an UnsupportedOperationException.

Implements `decaf::io::InputStream` (p. 1633).

6.636.3.12 `virtual bool decaf::net::SocketInputStream::tryLock () throw (decaf::lang::exceptions::RuntimeException)` [inline, virtual]

Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

Returns

true if the lock was acquired, false if it is already held by another thread.

Exceptions

RuntimeException if an error occurs while locking the object.

Implements `decaf::util::concurrent::Synchronizable` (p. 2935).

6.636.3.13 virtual void decaf::net::SocketInputStream::unlock () throw (decaf::lang::exceptions::RuntimeException) [inline, virtual]

Unlocks the object.

Exceptions

RuntimeException if an error occurs while unlocking the object.

Implements decaf::util::concurrent::Synchronizable (p. 2936).

6.636.3.14 virtual void decaf::net::SocketInputStream::wait (long long *millisecs*, int *nanos*) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException) [inline, virtual]

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

Parameters

millisecs the time in milliseconds to wait, or WAIT_INFINITE

nanos additional time in nanoseconds with a range of 0-999999

Exceptions

IllegalArgumentException if an error occurs or the nanos argument is not in the range of [0-999999]

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements decaf::util::concurrent::Synchronizable (p. 2940).

6.636.3.15 virtual void decaf::net::SocketInputStream::wait (long long *millisecs*) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException) [inline, virtual]

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval.

Parameters

millisecs the time in milliseconds to wait, or WAIT_INFINITE

Exceptions

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2938).

```
6.636.3.16  virtual void decaf::net::SocketInputStream::wait (    )
              throw ( decaf::lang::exceptions::RuntimeException,
                      decaf::lang::exceptions::IllegalMonitorStateException,
                      decaf::lang::exceptions::InterruptedException ) [inline, virtual]
```

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling.

Exceptions

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2937).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/SocketInputStream.h`

6.637 decaf::net::SocketOutputStream Class Reference

Output stream for performing write operations on a socket.

```
#include <src/main/decaf/net/SocketOutputStream.h>
```

Inheritance diagram for `decaf::net::SocketOutputStream`:

Public Member Functions

- **SocketOutputStream** (**Socket::SocketHandle** socket)
Constructor.
- virtual **~SocketOutputStream** ()
- virtual void **write** (unsigned char c) throw (io::IOException)
Writes a single byte to the output stream.
- virtual void **write** (const std::vector< unsigned char > &buffer) throw (io::IOException)

Writes an array of bytes to the output stream.

- virtual void **write** (const unsigned char *buffer, std::size_t offset, std::size_t len) throw (io::IOException, lang::exceptions::NullPointerException)

Writes an array of bytes to the output stream.

- virtual void **flush** () throw (io::IOException)

Flush - does nothing.

- virtual void **close** () throw (decaf::io::IOException)

Close - does nothing.

- virtual void **lock** () throw (decaf::lang::exceptions::RuntimeException)

Locks the object.

- virtual bool **tryLock** () throw (decaf::lang::exceptions::RuntimeException)

Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

- virtual void **unlock** () throw (decaf::lang::exceptions::RuntimeException)

Unlocks the object.

- virtual void **wait** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)

Waits on a signal from this object, which is generated by a call to Notify.

- virtual void **wait** (long long millisecs) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)

Waits on a signal from this object, which is generated by a call to Notify.

- virtual void **wait** (long long millisecs, int nanos) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)

Waits on a signal from this object, which is generated by a call to Notify.

- virtual void **notify** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)

Signals a waiter on this object that it can now wake up and continue.

- virtual void **notifyAll** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)

Signals the waiters on this object that it can now wake up and continue.

6.637.1 Detailed Description

Output stream for performing write operations on a socket.

6.637.2 Constructor & Destructor Documentation

6.637.2.1 `decaf::net::SocketOutputStream::SocketOutputStream (Socket::SocketHandle socket)`

Constructor.

Parameters

socket the socket handle.

6.637.2.2 `virtual decaf::net::SocketOutputStream::~~SocketOutputStream ()` [virtual]

6.637.3 Member Function Documentation

6.637.3.1 `virtual void decaf::net::SocketOutputStream::close () throw (decaf::io::IOException)` [virtual]

Close - does nothing.

It is the responsibility of the owner of the socket object to close it.

Exceptions

IOException

6.637.3.2 `virtual void decaf::net::SocketOutputStream::flush () throw (io::IOException)` [inline, virtual]

Flush - does nothing.

Exceptions

IOException

Implements `decaf::io::OutputStream` (p. 2317).

6.637.3.3 `virtual void decaf::net::SocketOutputStream::lock () throw (decaf::lang::exceptions::RuntimeException)` [inline, virtual]

Locks the object.

Exceptions

RuntimeException if an error occurs while locking the object.

Implements `decaf::util::concurrent::Synchronizable` (p. 2932).

6.637.3.4 virtual void decaf::net::SocketOutputStream::notify () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException) [inline, virtual]

Signals a waiter on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

RuntimeException if an error occurs while notifying one of the waiting threads.

Implements decaf::util::concurrent::Synchronizable (p. 2933).

6.637.3.5 virtual void decaf::net::SocketOutputStream::notifyAll () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException) [inline, virtual]

Signals the waiters on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

RuntimeException if an error occurs while notifying the waiting threads.

Implements decaf::util::concurrent::Synchronizable (p. 2934).

6.637.3.6 virtual bool decaf::net::SocketOutputStream::tryLock () throw (decaf::lang::exceptions::RuntimeException) [inline, virtual]

Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

Returns

true if the lock was acquired, false if it is already held by another thread.

Exceptions

RuntimeException if an error occurs while locking the object.

Implements decaf::util::concurrent::Synchronizable (p. 2935).

6.637.3.7 virtual void decaf::net::SocketOutputStream::unlock () throw (decaf::lang::exceptions::RuntimeException) [inline, virtual]

Unlocks the object.

Exceptions

RuntimeException if an error occurs while unlocking the object.

Implements `decaf::util::concurrent::Synchronizable` (p. 2936).

6.637.3.8 `virtual void decaf::net::SocketOutputStream::wait ()
throw (decaf::lang::exceptions::RuntimeException,
decaf::lang::exceptions::IllegalMonitorStateException,
decaf::lang::exceptions::InterruptedException) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling.

Exceptions

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements `decaf::util::concurrent::Synchronizable` (p. 2937).

6.637.3.9 `virtual void decaf::net::SocketOutputStream::wait (long long
milliseconds) throw (decaf::lang::exceptions::RuntimeException,
decaf::lang::exceptions::IllegalMonitorStateException,
decaf::lang::exceptions::InterruptedException) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval.

Parameters

milliseconds the time in milliseconds to wait, or WAIT_INFINITE

Exceptions

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements `decaf::util::concurrent::Synchronizable` (p. 2938).

6.637.3.10 `virtual void decaf::net::SocketOutputStream::wait (long long milliseconds,
int nanos) throw (decaf::lang::exceptions::RuntimeException,
decaf::lang::exceptions::IllegalArgumentOutOfRangeException,
decaf::lang::exceptions::IllegalMonitorStateException,
decaf::lang::exceptions::InterruptedException) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

Parameters

milliseconds the time in milliseconds to wait, or WAIT_INFINITE

nanos additional time in nanoseconds with a range of 0-999999

Exceptions

IllegalArgumentException if an error occurs or the nanos argument is not in the range of [0-999999]

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2940).

6.637.3.11 virtual void decaf::net::SocketOutputStream::write (const std::vector< unsigned char > & *buffer*) throw (io::IOException) [virtual]

Writes an array of bytes to the output stream.

Parameters

buffer The bytes to write.

Exceptions

IOException thrown if an error occurs.

Implements **decaf::io::OutputStream** (p. 2317).

6.637.3.12 virtual void decaf::net::SocketOutputStream::write (unsigned char *c*) throw (io::IOException) [virtual]

Writes a single byte to the output stream.

Parameters

c the byte.

Exceptions

IOException thrown if an error occurs.

Implements **decaf::io::OutputStream** (p. 2318).

6.637.3.13 `virtual void decaf::net::SocketOutputStream::write (const unsigned char * buffer, std::size_t offset, std::size_t len) throw (io::IOException, lang::exceptions::NullPointerException) [virtual]`

Writes an array of bytes to the output stream.

Parameters

- buffer* The array of bytes to write.
- offset* the position to start writing in buffer.
- len* The number of bytes from the buffer to be written.

Exceptions

- IOException* thrown if an error occurs.
- NullPointerException* thrown if buffer is Null.

Implements **decaf::io::OutputStream** (p. 2317).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/SocketOutputStream.h`

6.638 decaf::net::SocketTimeoutException Class Reference

```
#include <src/main/decaf/net/SocketTimeoutException.h>
```

Inheritance diagram for decaf::net::SocketTimeoutException:

Public Member Functions

- **SocketTimeoutException** () throw ()
Default Constructor.
- **SocketTimeoutException** (const Exception &ex) throw ()
Conversion Constructor from some other Exception.
- **SocketTimeoutException** (const **SocketTimeoutException** &ex) throw ()
Copy Constructor.
- **SocketTimeoutException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **SocketTimeoutException** (const std::exception *cause) throw ()
Constructor.
- **SocketTimeoutException** (const char *file, const int lineNumber, const char *msg,...) throw ()

Constructor - Initializes the file name and line number where this message occurred.

- virtual **SocketTimeoutException * clone () const**
Clones this exception.
- virtual **~SocketTimeoutException () throw ()**

6.638.1 Constructor & Destructor Documentation

6.638.1.1 decaf::net::SocketTimeoutException::SocketTimeoutException () throw () [inline]

Default Constructor.

6.638.1.2 decaf::net::SocketTimeoutException::SocketTimeoutException (const Exception & *ex*) throw () [inline]

Conversion Constructor from some other Exception.

Parameters

ex An exception that should become this type of Exception

References decaf::lang::Exception::Exception().

6.638.1.3 decaf::net::SocketTimeoutException::SocketTimeoutException (const SocketTimeoutException & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex An exception that should become this type of Exception

References decaf::lang::Exception::Exception().

6.638.1.4 decaf::net::SocketTimeoutException::SocketTimeoutException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.638.1.5 `decaf::net::SocketTimeoutException::SocketTimeoutException (const std::exception * cause) throw () [inline]`

Constructor.

Parameters

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.638.1.6 `decaf::net::SocketTimeoutException::SocketTimeoutException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.638.1.7 `virtual decaf::net::SocketTimeoutException::~~SocketTimeoutException () throw () [inline, virtual]`

6.638.2 Member Function Documentation

6.638.2.1 `virtual SocketTimeoutException* decaf::net::SocketTimeoutException::clone () const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

a new Exception instance that is a copy of this Exception object.

Reimplemented from `decaf::io::InterruptedIOException` (p.1695).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/SocketTimeoutException.h`

6.639 activemq::commands::BrokerError::StackTraceElement Struct Reference

```
#include <src/main/activemq/commands/BrokerError.h>
```

Data Fields

- std::string **ClassName**
- std::string **FileName**
- std::string **MethodName**
- int **LineNumber**

6.639.1 Field Documentation

6.639.1.1 std::string **activemq::commands::BrokerError::StackTraceElement::ClassName**

6.639.1.2 std::string **activemq::commands::BrokerError::StackTraceElement::FileName**

6.639.1.3 int **activemq::commands::BrokerError::StackTraceElement::LineNumber**

6.639.1.4 std::string **activemq::commands::BrokerError::StackTraceElement::MethodName**

The documentation for this struct was generated from the following file:

- src/main/activemq/commands/**BrokerError.h**

6.640 decaf::internal::io::StandardErrorOutputStream Class Reference

Wrapper Around the Standard error Output facility on the current platform.

```
#include <src/main/decaf/internal/io/StandardErrorOutputStream.h>
```

Inheritance diagram for decaf::internal::io::StandardErrorOutputStream:

Public Member Functions

- **StandardErrorOutputStream ()**
Default Constructor.
- virtual **~StandardErrorOutputStream ()**
- virtual void **write** (unsigned char c) throw (decaf::io::IOException)
Writes a single byte to the output stream.
- virtual void **write** (const std::vector< unsigned char > &buffer) throw (decaf::io::IOException)
Writes an array of bytes to the output stream.
- virtual void **write** (const unsigned char *buffer, std::size_t offset, std::size_t len) throw (decaf::io::IOException, lang::exceptions::NullPointerException)

Writes an array of bytes to the output stream.

- virtual void **flush** () throw (decaf::io::IOException)

Invokes flush on the target output stream.

- virtual void **close** () throw (decaf::io::IOException)

Invokes close on the target output stream.

- virtual void **lock** () throw (decaf::lang::exceptions::RuntimeException)

Locks the object.

- virtual bool **tryLock** () throw (decaf::lang::exceptions::RuntimeException)

Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

- virtual void **unlock** () throw (decaf::lang::exceptions::RuntimeException)

Unlocks the object.

- virtual void **wait** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)

Waits on a signal from this object, which is generated by a call to Notify.

- virtual void **wait** (long long millisecs) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)

Waits on a signal from this object, which is generated by a call to Notify.

- virtual void **wait** (long long millisecs, int nanos) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)

Waits on a signal from this object, which is generated by a call to Notify.

- virtual void **notify** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)

Signals a waiter on this object that it can now wake up and continue.

- virtual void **notifyAll** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)

Signals the waiters on this object that it can now wake up and continue.

6.640.1 Detailed Description

Wrapper Around the Standard error Output facility on the current platform. This allows for the use of alternate output methods on platforms or compilers that do not support `std::cerr`.

6.640.2 Constructor & Destructor Documentation

6.640.2.1 decaf::internal::io::StandardErrorOutputStream::StandardErrorOutputStream ()

Default Constructor.

6.640.2.2 virtual decaf::internal::io::StandardErrorOutputStream::~~StandardErrorOutputStream () [virtual]

6.640.3 Member Function Documentation

6.640.3.1 virtual void decaf::internal::io::StandardErrorOutputStream::close () throw (decaf::io::IOException) [inline, virtual]

Invokes close on the target output stream.

throws IOException if an error occurs

6.640.3.2 virtual void decaf::internal::io::StandardErrorOutputStream::flush () throw (decaf::io::IOException) [virtual]

Invokes flush on the target output stream.

throws **decaf::io::IOException** (p. 1707) if an error occurs

Implements **decaf::io::OutputStream** (p. 2317).

6.640.3.3 virtual void decaf::internal::io::StandardErrorOutputStream::lock () throw (decaf::lang::exceptions::RuntimeException) [inline, virtual]

Locks the object.

Exceptions

RuntimeException if an error occurs while locking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2932).

6.640.3.4 virtual void decaf::internal::io::StandardErrorOutputStream::notify () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException) [inline, virtual]

Signals a waiter on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

RuntimeException if an error occurs while notifying one of the waiting threads.

Implements `decaf::util::concurrent::Synchronizable` (p. 2933).

6.640.3.5 `virtual void decaf::internal::io::StandardErrorOutputStream::notifyAll () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException) [inline, virtual]`

Signals the waiters on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

RuntimeException if an error occurs while notifying the waiting threads.

Implements `decaf::util::concurrent::Synchronizable` (p. 2934).

6.640.3.6 `virtual bool decaf::internal::io::StandardErrorOutputStream::tryLock () throw (decaf::lang::exceptions::RuntimeException) [inline, virtual]`

Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

Returns

true if the lock was acquired, false if it is already held by another thread.

Exceptions

RuntimeException if an error occurs while locking the object.

Implements `decaf::util::concurrent::Synchronizable` (p. 2935).

6.640.3.7 `virtual void decaf::internal::io::StandardErrorOutputStream::unlock () throw (decaf::lang::exceptions::RuntimeException) [inline, virtual]`

Unlocks the object.

Exceptions

RuntimeException if an error occurs while unlocking the object.

Implements `decaf::util::concurrent::Synchronizable` (p. 2936).

6.640.3.8 `virtual void decaf::internal::io::StandardErrorOutputStream::wait () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling.

Exceptions

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2937).

6.640.3.9 `virtual void decaf::internal::io::StandardErrorOutputStream::wait (long long millisecs) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval.

Parameters

millisecs the time in milliseconds to wait, or WAIT_INFINITE

Exceptions

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2938).

6.640.3.10 `virtual void decaf::internal::io::StandardErrorOutputStream::wait (long long millisecs, int nanos) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

Parameters

millisecs the time in milliseconds to wait, or WAIT_INFINITE

nanos additional time in nanoseconds with a range of 0-999999

Exceptions

IllegalArgumentException if an error occurs or the *nanos* argument is not in the range of [0-999999]

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2940).

6.640.3.11 `virtual void decaf::internal::io::StandardErrorOutputStream::write (const std::vector< unsigned char > & buffer) throw (decaf::io::IOException) [virtual]`

Writes an array of bytes to the output stream.

Parameters

buffer The bytes to write.

Exceptions

IOException thrown if an error occurs.

Implements **decaf::io::OutputStream** (p. 2317).

6.640.3.12 `virtual void decaf::internal::io::StandardErrorOutputStream::write (unsigned char c) throw (decaf::io::IOException) [virtual]`

Writes a single byte to the output stream.

Parameters

c the byte.

Exceptions

IOException thrown if an error occurs.

Implements **decaf::io::OutputStream** (p. 2318).

6.640.3.13 `virtual void decaf::internal::io::StandardErrorOutputStream::write (const unsigned char * buffer, std::size_t offset, std::size_t len) throw (decaf::io::IOException, lang::exceptions::NullPointerException) [virtual]`

Writes an array of bytes to the output stream.

Parameters

- buffer* The array of bytes to write.
- offset* The position to start writing in buffer.
- len* The number of bytes from the buffer to be written.

Exceptions

- decaf::io::IOException* (p. 1707) thrown if an error occurs.
- NullPointerException* if buffer is null.

Implements **decaf::io::OutputStream** (p. 2317).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/io/**StandardErrorOutputStream.h**

6.641 decaf::internal::io::StandardInputStream Class Reference

```
#include <src/main/decaf/internal/io/StandardInputStream.h>
```

Inheritance diagram for decaf::internal::io::StandardInputStream:

Public Member Functions

- **StandardInputStream** ()
- virtual **~StandardInputStream** ()
- virtual std::size_t **available** () const throw (decaf::io::IOException)
Indicates the number of bytes available.
- virtual int **read** () throw (decaf::io::IOException)
Reads a single byte from the buffer.
- virtual int **read** (unsigned char *buffer, std::size_t offset, std::size_t bufferSize) throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException)
Reads an array of bytes from the buffer.
- virtual void **close** () throw (decaf::io::IOException)
Closes the target input stream.
- virtual std::size_t **skip** (std::size_t num) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)
Skips over and discards n bytes of data from this input stream.
- virtual void **mark** (int readLimit DECAF_UNUSED)
Marks the current position in the stream A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.

- virtual void **reset** () throw (decaf::io::IOException)

Repositions this stream to the position at the time the mark method was last called on this input stream.

- virtual bool **markSupported** () const

Determines if this input stream supports the mark and reset methods.

Protected Member Functions

- virtual void **lock** () throw (decaf::lang::exceptions::RuntimeException)

Locks the object.

- virtual bool **tryLock** () throw (decaf::lang::exceptions::RuntimeException)

Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

- virtual void **unlock** () throw (decaf::lang::exceptions::RuntimeException)

Unlocks the object.

- virtual void **wait** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)

Waits on a signal from this object, which is generated by a call to Notify.

- virtual void **wait** (long long millisecs) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)

Waits on a signal from this object, which is generated by a call to Notify.

- virtual void **wait** (long long millisecs, int nanos) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentOutOfRangeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)

Waits on a signal from this object, which is generated by a call to Notify.

- virtual void **notify** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)

Signals a waiter on this object that it can now wake up and continue.

- virtual void **notifyAll** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)

Signals the waiters on this object that it can now wake up and continue.

6.641.1 Constructor & Destructor Documentation

6.641.1.1 decaf::internal::io::StandardInputStream::StandardInputStream ()

6.641.1.2 virtual decaf::internal::io::StandardInputStream::~~StandardInputStream () [virtual]

6.641.2 Member Function Documentation

6.641.2.1 virtual std::size_t decaf::internal::io::StandardInputStream::available () const throw (decaf::io::IOException) [virtual]

Indicates the number of bytes available.

Returns

The number of bytes until the end of the internal buffer.

Implements **decaf::io::InputStream** (p. 1631).

6.641.2.2 virtual void decaf::internal::io::StandardInputStream::close () throw (decaf::io::IOException) [inline, virtual]

Closes the target input stream.

Exceptions

IOException thrown if an error occurs.

6.641.2.3 virtual void decaf::internal::io::StandardInputStream::lock () throw (decaf::lang::exceptions::RuntimeException) [inline, protected, virtual]

Locks the object.

Exceptions

RuntimeException if an error occurs while locking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2932).

6.641.2.4 virtual void decaf::internal::io::StandardInputStream::mark (int readLimit *DECAF_UNUSED*) [inline, virtual]

Marks the current position in the stream. A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.

If a stream instance reports that marks are supported then the stream will ensure that the same bytes can be read again after the reset method is called so long the readLimit is not reached.

Parameters

readLimit - max bytes read before marked position is invalid.

Implements **decaf::io::InputStream** (p. 1631).

6.641.2.5 `virtual bool decaf::internal::io::StandardInputStream::markSupported () const [inline, virtual]`

Determines if this input stream supports the mark and reset methods.

Whether or not mark and reset are supported is an invariant property of a particular input stream instance.

Returns

true if this stream instance supports marks

Implements **decaf::io::InputStream** (p. 1631).

6.641.2.6 `virtual void decaf::internal::io::StandardInputStream::notify () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException) [inline, protected, virtual]`

Signals a waiter on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

RuntimeException if an error occurs while notifying one of the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 2933).

6.641.2.7 `virtual void decaf::internal::io::StandardInputStream::notifyAll () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException) [inline, protected, virtual]`

Signals the waiters on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

RuntimeException if an error occurs while notifying the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 2934).

6.641.2.8 `virtual int decaf::internal::io::StandardInputStream::read () throw (decaf::io::IOException) [virtual]`

Reads a single byte from the buffer.

Returns

The next byte.

Exceptions

IOException thrown if an error occurs.

Implements **decaf::io::InputStream** (p. 1632).

6.641.2.9 `virtual int decaf::internal::io::StandardInputStream::read (unsigned char * buffer, std::size_t offset, std::size_t bufferSize) throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException) [virtual]`

Reads an array of bytes from the buffer.

Parameters

buffer (out) the target buffer.

offset the position in the buffer to start reading from.

bufferSize the size of the output buffer.

Returns

The number of bytes read.

Exceptions

IOException thrown if an error occurs.

NullPointerException if buffer is null.

Implements **decaf::io::InputStream** (p. 1632).

6.641.2.10 `virtual void decaf::internal::io::StandardInputStream::reset () throw (decaf::io::IOException) [virtual]`

Repositions this stream to the position at the time the mark method was last called on this input stream.

If the method markSupported returns true, then: * If the method mark has not been called since the stream was created, or the number of bytes read from the stream since mark was last called is larger than the argument to mark at that last call, then an IOException might be thrown. * If such an IOException is not thrown, then the stream is reset to a state such that all the bytes read since the most recent call to mark (or since the start of the file, if mark has not been called) will be resupplied to subsequent callers of the read method, followed by any bytes that otherwise would have been the next input data as of the time of the call to reset. If the method markSupported returns false, then: * The call to reset may throw an IOException. * If an IOException is not thrown, then the stream is reset to a fixed state that depends on the particular type of the input stream and how it was created. The bytes that will be supplied to subsequent callers of the read method depend on the particular type of the input stream.

Exceptions

IOException

Implements **decaf::io::InputStream** (p. 1633).

6.641.2.11 `virtual std::size_t decaf::internal::io::StandardInputStream::skip
(std::size_t num) throw (decaf::io::IOException,
decaf::lang::exceptions::UnsupportedOperationException) [virtual]`

Skips over and discards *n* bytes of data from this input stream.

The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before *n* bytes have been skipped is only one possibility. The actual number of bytes skipped is returned. If *n* is negative, no bytes are skipped.

The skip method of `InputStream` creates a byte array and then repeatedly reads into it until *n* bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.

Parameters

num - the number of bytes to skip

Returns

total bytes skipped

Exceptions

IOException if an error occurs

UnsupportedOperationException If skip is not supported.

Implements `decaf::io::InputStream` (p. 1633).

6.641.2.12 `virtual bool decaf::internal::io::StandardInputStream::tryLock ()
throw (decaf::lang::exceptions::RuntimeException) [inline,
protected, virtual]`

Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

Returns

true if the lock was acquired, false if it is already held by another thread.

Exceptions

RuntimeException if an error occurs while locking the object.

Implements `decaf::util::concurrent::Synchronizable` (p. 2935).

6.641.2.13 `virtual void decaf::internal::io::StandardInputStream::unlock () throw
(decaf::lang::exceptions::RuntimeException) [inline, protected,
virtual]`

Unlocks the object.

Exceptions

RuntimeException if an error occurs while unlocking the object.

Implements `decaf::util::concurrent::Synchronizable` (p. 2936).

```

6.641.2.14  virtual void decaf::internal::io::StandardInputStream::wait
            ( long long millisecs, int nanos ) throw
            ( decaf::lang::exceptions::RuntimeException,
              decaf::lang::exceptions::IllegalArgumentException,
              decaf::lang::exceptions::IllegalMonitorStateException,
              decaf::lang::exceptions::InterruptedException ) [inline, protected,
              virtual]

```

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

Parameters

millisecs the time in milliseconds to wait, or WAIT_INFINITE

nanos additional time in nanoseconds with a range of 0-999999

Exceptions

IllegalArgumentException if an error occurs or the nanos argument is not in the range of [0-999999]

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2940).

```

6.641.2.15  virtual void decaf::internal::io::StandardInputStream::wait ( long
            long millisecs ) throw ( decaf::lang::exceptions::RuntimeException,
            decaf::lang::exceptions::IllegalMonitorStateException,
            decaf::lang::exceptions::InterruptedException ) [inline, protected,
            virtual]

```

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval.

Parameters

millisecs the time in milliseconds to wait, or WAIT_INFINITE

Exceptions

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2938).

6.641.2.16 `virtual void decaf::internal::io::StandardInputStream::wait () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException) [inline, protected, virtual]`

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling.

Exceptions

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements `decaf::util::concurrent::Synchronizable` (p. 2937).

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/io/StandardInputStream.h`

6.642 decaf::internal::io::StandardOutputStream Class Reference

```
#include <src/main/decaf/internal/io/StandardOutputStream.h>
```

Inheritance diagram for `decaf::internal::io::StandardOutputStream`:

Public Member Functions

- `StandardOutputStream ()`
- `virtual ~StandardOutputStream ()`
- `virtual void write (unsigned char c) throw (decaf::io::IOException)`
Writes a single byte to the output stream.
- `virtual void write (const std::vector< unsigned char > &buffer) throw (decaf::io::IOException)`
Writes an array of bytes to the output stream.
- `virtual void write (const unsigned char *buffer, std::size_t offset, std::size_t len) throw (decaf::io::IOException, lang::exceptions::NullPointerException)`
Writes an array of bytes to the output stream.
- `virtual void flush () throw (decaf::io::IOException)`
Invokes flush on the target output stream.
- `virtual void close () throw (decaf::io::IOException)`

Invokes close on the target output stream.

- virtual void **lock** () throw (decaf::lang::exceptions::RuntimeException)
Locks the object.
- virtual bool **tryLock** () throw (decaf::lang::exceptions::RuntimeException)
Attempts to Lock the object, if the lock is already held by another thread than this method returns false.
- virtual void **unlock** () throw (decaf::lang::exceptions::RuntimeException)
Unlocks the object.
- virtual void **wait** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (long long millisecs) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (long long millisecs, int nanos) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **notify** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)
Signals a waiter on this object that it can now wake up and continue.
- virtual void **notifyAll** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)
Signals the waiters on this object that it can now wake up and continue.

6.642.1 Constructor & Destructor Documentation

6.642.1.1 decaf::internal::io::StandardOutputStream::StandardOutputStream ()

6.642.1.2 virtual
decaf::internal::io::StandardOutputStream::~~StandardOutputStream () [virtual]

6.642.2 Member Function Documentation

6.642.2.1 virtual void decaf::internal::io::StandardOutputStream::close () throw (decaf::io::IOException) [inline, virtual]

Invokes close on the target output stream.

throws IOException if an error occurs

6.642.2.2 `virtual void decaf::internal::io::StandardOutputStream::flush () throw (decaf::io::IOException) [virtual]`

Invokes flush on the target output stream.

throws `IOException` if an error occurs

Implements `decaf::io::OutputStream` (p. 2317).

6.642.2.3 `virtual void decaf::internal::io::StandardOutputStream::lock () throw (decaf::lang::exceptions::RuntimeException) [inline, virtual]`

Locks the object.

Exceptions

RuntimeException if an error occurs while locking the object.

Implements `decaf::util::concurrent::Synchronizable` (p. 2932).

6.642.2.4 `virtual void decaf::internal::io::StandardOutputStream::notify () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException) [inline, virtual]`

Signals a waiter on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

RuntimeException if an error occurs while notifying one of the waiting threads.

Implements `decaf::util::concurrent::Synchronizable` (p. 2933).

6.642.2.5 `virtual void decaf::internal::io::StandardOutputStream::notifyAll () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException) [inline, virtual]`

Signals the waiters on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

RuntimeException if an error occurs while notifying the waiting threads.

Implements `decaf::util::concurrent::Synchronizable` (p. 2934).

6.642.2.6 virtual bool decaf::internal::io::StandardOutputStream::tryLock ()
throw (decaf::lang::exceptions::RuntimeException) [inline, virtual]

Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

Returns

true if the lock was acquired, false if it is already held by another thread.

Exceptions

RuntimeException if an error occurs while locking the object.

Implements decaf::util::concurrent::Synchronizable (p. 2935).

6.642.2.7 virtual void decaf::internal::io::StandardOutputStream::unlock ()
throw (decaf::lang::exceptions::RuntimeException) [inline, virtual]

Unlocks the object.

Exceptions

RuntimeException if an error occurs while unlocking the object.

Implements decaf::util::concurrent::Synchronizable (p. 2936).

6.642.2.8 virtual void decaf::internal::io::StandardOutputStream::wait
() throw (decaf::lang::exceptions::RuntimeException,
decaf::lang::exceptions::IllegalMonitorStateException,
decaf::lang::exceptions::InterruptedException) [inline, virtual]

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling.

Exceptions

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements decaf::util::concurrent::Synchronizable (p. 2937).

6.642.2.9 virtual void decaf::internal::io::StandardOutputStream::wait (long
long *millisecs*) throw (decaf::lang::exceptions::RuntimeException,
decaf::lang::exceptions::IllegalMonitorStateException,
decaf::lang::exceptions::InterruptedException) [inline, virtual]

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval.

Parameters

milliseconds the time in milliseconds to wait, or WAIT_INFINITE

Exceptions

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2938).

```
6.642.2.10 virtual void decaf::internal::io::StandardOutputStream::wait
( long long milliseconds, int nanos ) throw
( decaf::lang::exceptions::RuntimeException,
  decaf::lang::exceptions::IllegalArgumentException,
  decaf::lang::exceptions::IllegalMonitorStateException,
  decaf::lang::exceptions::InterruptedException ) [inline, virtual]
```

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

Parameters

milliseconds the time in milliseconds to wait, or WAIT_INFINITE

nanos additional time in nanoseconds with a range of 0-999999

Exceptions

IllegalArgumentException if an error occurs or the nanos argument is not in the range of [0-999999]

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2940).

```
6.642.2.11 virtual void decaf::internal::io::StandardOutputStream::write ( const
std::vector< unsigned char > & buffer ) throw ( decaf::io::IOException
) [virtual]
```

Writes an array of bytes to the output stream.

Parameters

buffer The bytes to write.

Exceptions

IOException thrown if an error occurs.

Implements **decaf::io::OutputStream** (p. 2317).

6.642.2.12 virtual void decaf::internal::io::StandardOutputStream::write (unsigned char *c*) throw (decaf::io::IOException) [virtual]

Writes a single byte to the output stream.

Parameters

c the byte.

Exceptions

IOException thrown if an error occurs.

Implements **decaf::io::OutputStream** (p. 2318).

6.642.2.13 virtual void decaf::internal::io::StandardOutputStream::write (const unsigned char * *buffer*, std::size_t *offset*, std::size_t *len*) throw (decaf::io::IOException, lang::exceptions::NullPointerException) [virtual]

Writes an array of bytes to the output stream.

Parameters

buffer The array of bytes to write.

offset, the position to start writing in buffer.

len The number of bytes from the buffer to be written.

Exceptions

IOException thrown if an error occurs.

NullPointerException if buffer is null.

Implements **decaf::io::OutputStream** (p. 2317).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/io/**StandardOutputStream.h**

6.643 cms::Startable Class Reference

Interface for a class that implements the start method.

```
#include <src/main/cms/Startable.h>
```

Inheritance diagram for cms::Startable:

Public Member Functions

- virtual **~Startable** ()
- virtual void **start** ()=0 throw (CMSEException)
Starts the service.

6.643.1 Detailed Description

Interface for a class that implements the start method. An object that implements the **Startable** (p.2831) interface implies that until its start method is called it will be considered to be in a closed or stopped state and will throw an Exception to indicate that it is not in an started state if one of its methods is called.

Since

1.0

6.643.2 Constructor & Destructor Documentation

6.643.2.1 virtual cms::Startable::~~Startable () [inline, virtual]

6.643.3 Member Function Documentation

6.643.3.1 virtual void cms::Startable::start () throw (CMSEException) [pure virtual]

Starts the service.

Exceptions

CMSEException (p. 960) if an internal error occurs while starting.

The documentation for this class was generated from the following file:

- src/main/cms/Startable.h

6.644 decaf::lang::STATIC_CAST_TOKEN Struct Reference

```
#include <src/main/decaf/lang/Pointer.h>
```

The documentation for this struct was generated from the following file:

- `src/main/decaf/lang/Pointer.h`

6.645 activemq::core::ActiveMQConstants::StaticInitializer Class Reference

```
#include <src/main/activemq/core/ActiveMQConstants.h>
```

Public Member Functions

- `StaticInitializer ()`
- `virtual ~StaticInitializer ()`

Static Public Attributes

- `static std::string destOptions [NUM_OPTIONS]`
- `static std::string uriParams [NUM_PARAMS]`
- `static std::map< std::string, DestinationOption > destOptionMap`
- `static std::map< std::string, URIParam > uriParamsMap`

6.645.1 Constructor & Destructor Documentation

6.645.1.1 `activemq::core::ActiveMQConstants::StaticInitializer::StaticInitializer (`
`)`

6.645.1.2 `virtual`
`activemq::core::ActiveMQConstants::StaticInitializer::~~StaticInitializer (`
`) [inline, virtual]`

6.645.2 Field Documentation

6.645.2.1 `std::map<std::string, DestinationOption> ac-`
`tivemq::core::ActiveMQConstants::StaticInitializer::destOptionMap`
`[static]`

6.645.2.2 `std::string`
`activemq::core::ActiveMQConstants::StaticInitializer::destOptions[NUM_`
`OPTIONS] [static]`

6.645.2.3 `std::string`
`activemq::core::ActiveMQConstants::StaticInitializer::uriParams[NUM_`
`PARAMS] [static]`

6.645.2.4 `std::map<std::string, URIParam> ac-`
`tivemq::core::ActiveMQConstants::StaticInitializer::uriParamsMap`
`[static]`

The documentation for this class was generated from the following file:

- `src/main/activemq/core/ActiveMQConstants.h`

6.646 `decaf::util::StlList< E >` Class Template Reference

List (p.1865) class that wraps the STL list object to provide a simpler interface and additional methods not provided by the STL type.

```
#include <src/main/decaf/util/StlList.h>
```

Inheritance diagram for `decaf::util::StlList< E >`:

Data Structures

- class **ConstStlListIterator**
- class **StlListIterator**

Public Member Functions

- **StlList** ()
Default constructor - does nothing.
- **StlList** (const **StlList** &source)
Copy constructor - copies the content of the given set into this one.
- **StlList** (const **Collection**< E > &source)
Copy constructor - copies the content of the given set into this one.
- virtual **~StlList** ()
- virtual bool **equals** (const **StlList** &source) const
- virtual **Iterator**< E > * **iterator** ()
- virtual **Iterator**< E > * **iterator** () const
- virtual **ListIterator**< E > * **listIterator** ()
Returns
a list iterator over the elements in this list (in proper sequence).
- virtual **ListIterator**< E > * **listIterator** () const
- virtual **ListIterator**< E > * **listIterator** (std::size_t index) throw (decaf::lang::exceptions::IndexOutOfBoundsException)
Parameters
index index of first element to be returned from the list iterator (by a call to the next method).

Returns

a list iterator of the elements in this list (in proper sequence), starting at the specified position in this list. The specified index indicates the first element that would be returned by an initial call to next. An initial call to previous would return the element with the specified index minus one.

Exceptions

IndexOutOfBoundsException if the index is out of range ($\text{index} < 0 \parallel \text{index} > \text{size}()$ (p. 990))

- virtual **ListIterator**< E > * **listIterator** (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException)
- virtual void **copy** (const **StlList** &source)

- virtual void **clear** () throw (lang::exceptions::UnsupportedOperationException)

Removes all of the elements from this collection (optional operation).

The collection will be empty after this method returns.

This implementation iterates over this collection, removing each element using the **Iterator.remove** (p. 1718) operation. Most implementations will probably choose to override this method for efficiency.

Note that this implementation will throw an **UnsupportedOperationException** if the iterator returned by this collection's iterator method does not implement the remove method and this collection is non-empty.

Exceptions

UnsupportedOperationException if the clear operation is not supported by this collection

- virtual bool **contains** (const E &value) const throw (lang::Exception)

Returns true if this collection contains the specified element.

This implementation iterates over the elements in the collection, checking each element in turn for equality with the specified element.

Parameters

value - the value whose presence is to be queried for in this **Collection** (p. 982).

Returns

true if the value is contained in this collection

Exceptions

Exception if an error occurs,

- virtual std::size_t **indexOf** (const E &value) throw (decaf::lang::exceptions::NoSuchElementException)

Returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element.

More formally, returns the lowest index i such that $\text{get}(i) == \text{value}$, or -1 if there is no such index.

Parameters

value - element to search for

Returns

the index of the first occurrence of the specified element in this list,

Exceptions

NoSuchElementException if value is not in the list

- virtual std::size_t **lastIndexOf** (const E &value) throw (decaf::lang::exceptions::NoSuchElementException)

Returns the index of the last occurrence of the specified element in this list, or -1 if this list does not contain the element.

More formally, returns the highest index i such that $get(i) == value$ or -1 if there is no such index.

Parameters

value - element to search for

Returns

the index of the last occurrence of the specified element in this list.

Exceptions

***NoSuchElementException** if value is not in the list*

- virtual bool **isEmpty** () const

Returns true if this collection contains no elements.

*This implementation returns **size()** (p. 990) == 0.*

Returns

true if the size method return 0.

- virtual std::size_t **size** () const

Returns the number of elements in this collection.

If this collection contains more than Integer.MAX_VALUE elements, returns Integer.MAX_VALUE.

Returns

the number of elements in this collection

- virtual E **get** (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException)

Gets the element contained at position passed.

Parameters

index - position to get

Returns

value at index

- virtual E **set** (std::size_t index, const E &element) throw (decaf::lang::exceptions::IndexOutOfBoundsException)

Replaces the element at the specified position in this list with the specified element.

Parameters

index - index of the element to replace

element - element to be stored at the specified position

Returns

the element previously at the specified position

Exceptions

***IndexOutOfBoundsException** - if the index is greater than size*

- virtual bool **add** (const E &value) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException)

Returns true if this collection changed as a result of the call.

(Returns false if this collection does not permit duplicates and already contains the specified element.)

Collections that support this operation may place limitations on what elements may be added to this collection. In particular, some collections will refuse to add null elements, and others will impose restrictions on the type of elements that may be added. **Collection** (p. 982) classes should clearly specify in their documentation any restrictions on what elements may be added.

If a collection refuses to add a particular element for any reason other than that it already contains the element, it must throw an exception (rather than returning false). This preserves the invariant that a collection always contains the specified element after this call returns.

For non-pointer values, i.e. class instances or string's the object will be copied into the collection, thus the object must support being copied, must not hide the copy constructor and assignment operator.

Parameters

value - reference to the element to add.

Returns

true if the element was added

Exceptions

UnsupportedOperationException

IllegalArgumentException

IllegalStateException if the element cannot be added at this time due to insertion restrictions

- virtual void **add** (std::size_t index, const E &element) throw (decaf::lang::exceptions::UnsupportedOperationException, lang::exceptions::IndexOutOfBoundsException)

Inserts the specified element at the specified position in this list.

Shifts the element currently at that position (if any) and any subsequent elements to the right (adds one to their indices).

Parameters

index - index at which the specified element is to be inserted

element - element to be inserted

Exceptions

IndexOutOfBoundsException - if the index is greater than size

UnsupportedOperationException - If the collection is non-modifiable.

- virtual bool **addAll** (std::size_t index, const **Collection**< E > &source) throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IndexOutOfBoundsException)

Inserts all of the elements in the specified collection into this list at the specified position (optional operation).

Shifts the element currently at that position (if any) and any subsequent elements to the right (increases their indices). The new elements will appear in this list in the order that they are returned by the specified collection's iterator. The behavior of this operation is undefined if the specified collection is modified while the operation is in progress. (Note that this will occur if the specified collection is this list, and it's nonempty.)

Parameters

index The index at which to insert the first element from the specified collection

source The **Collection** (p. 982) containing elements to be added to this list

Returns

true if this list changed as a result of the call

Exceptions

IndexOutOfBoundsException - if the index is greater than size

UnsupportedOperationException - If the collection is non-modifiable.

- virtual bool **remove** (const E &value) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException)

Removes a single instance of the specified element from this collection, if it is present (optional operation).

*More formally, removes the first element *e* such that *e* == *o*, if this collection contains one or more such elements. Returns true if this collection contained the specified element (or equivalently, if this collection changed as a result of the call).*

This implementation iterates over the collection looking for the specified element. If it finds the element, it removes the element from the collection using the iterator's remove method.

Note that this implementation throws an `UnsupportedOperationException` if the iterator returned by this collection's iterator method does not implement the remove method and this collection contains the specified object.

Parameters

value - element to be removed from this collection, if present

Returns

true if an element was removed as a result of this call

Exceptions

UnsupportedOperationException if the remove operation is not supported by this collection.

IllegalArgumentException If the value is not a valid entry for this **Collection** (p. 982).

- virtual E **remove** (std::size_t index) throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IndexOutOfBoundsException)

Removes the element at the specified position in this list.

Shifts any subsequent elements to the left (subtracts one from their indices). Returns the element that was removed from the list.

Parameters

index - the index of the element to be removed

Returns

the element previously at the specified position

Exceptions

IndexOutOfBoundsException - if the index is greater than size

UnsupportedOperationException - If the collection is non-modifiable.

6.646.1 Detailed Description

```
template<typename E> class decaf::util::StlList< E >
```

List (p. 1865) class that wraps the STL list object to provide a simpler interface and additional methods not provided by the STL type.

6.646.2 Constructor & Destructor Documentation

6.646.2.1 `template<typename E> decaf::util::StlList< E >::StlList () [inline]`

Default constructor - does nothing.

6.646.2.2 `template<typename E> decaf::util::StlList< E >::StlList (const StlList< E > & source) [inline]`

Copy constructor - copies the content of the given set into this one.

Parameters

source The source set.

6.646.2.3 `template<typename E> decaf::util::StlList< E >::StlList (const Collection< E > & source) [inline]`

Copy constructor - copies the content of the given set into this one.

Parameters

source The source set.

6.646.2.4 `template<typename E> virtual decaf::util::StlList< E >::~~StlList () [inline, virtual]`

6.646.3 Member Function Documentation

6.646.3.1 `template<typename E> virtual bool decaf::util::StlList< E >::add (const E & value) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException) [inline, virtual]`

Returns true if this collection changed as a result of the call.

(Returns false if this collection does not permit duplicates and already contains the specified element.)

Collections that support this operation may place limitations on what elements may be added to this collection. In particular, some collections will refuse to add null elements, and others will impose restrictions on the type of elements that may be added. **Collection** (p. 982) classes should clearly specify in their documentation any restrictions on what elements may be added.

If a collection refuses to add a particular element for any reason other than that it already contains the element, it must throw an exception (rather than returning false). This preserves the invariant that a collection always contains the specified element after this call returns.

For non-pointer values, i.e. class instances or string's the object will be copied into the collection, thus the object must support being copied, must not hide the copy constructor and assignment operator.

Parameters

value - reference to the element to add.

Returns

true if the element was added

Exceptions*UnsupportedOperationException**IllegalArgumentException**IllegalStateException* if the element cannot be added at this time due to insertion restrictionsImplements **decaf::util::Collection< E >** (p. 984).Referenced by **decaf::util::StlList< Pointer< BackupTransport > >::addAll()**.

6.646.3.2 `template<typename E> virtual void decaf::util::StlList< E >::add (std::size_t index, const E & element)
 throw (lang::exceptions::UnsupportedOperationException,
 lang::exceptions::IndexOutOfBoundsException) [inline, virtual]`

Inserts the specified element at the specified position in this list.

Shifts the element currently at that position (if any) and any subsequent elements to the right (adds one to their indices).

Parameters*index* - index at which the specified element is to be inserted*element* - element to be inserted**Exceptions***IndexOutOfBoundsException* - if the index is greater than size*UnsupportedOperationException* - If the collection is non-modifiable.Implements **decaf::util::List< E >** (p. 1867).

6.646.3.3 `template<typename E> virtual bool decaf::util::StlList< E >::addAll
 (std::size_t index, const Collection< E > & source)
 throw (decaf::lang::exceptions::UnsupportedOperationException,
 decaf::lang::exceptions::IndexOutOfBoundsException) [inline,
 virtual]`

Inserts all of the elements in the specified collection into this list at the specified position (optional operation).

Shifts the element currently at that position (if any) and any subsequent elements to the right (increases their indices). The new elements will appear in this list in the order that they are returned by the specified collection's iterator. The behavior of this operation is undefined if the specified collection is modified while the operation is in progress. (Note that this will occur if the specified collection is this list, and it's nonempty.)

Parameters*index* The index at which to insert the first element from the specified collection*source* The **Collection** (p. 982) containing elements to be added to this list

Returns

true if this list changed as a result of the call

Exceptions

IndexOutOfBoundsException - if the index is greater than size

UnsupportedOperationException - If the collection is non-modifiable.

Implements **decaf::util::List< E >** (p.1867).

6.646.3.4 `template<typename E> virtual void decaf::util::StlList< E >::clear ()
throw (lang::exceptions::UnsupportedOperationException) [inline,
virtual]`

Removes all of the elements from this collection (optional operation).

The collection will be empty after this method returns.

This implementation iterates over this collection, removing each element using the **Iterator.remove** (p.1718) operation. Most implementations will probably choose to override this method for efficiency.

Note that this implementation will throw an **UnsupportedOperationException** if the iterator returned by this collection's iterator method does not implement the remove method and this collection is non-empty.

Exceptions

UnsupportedOperationException if the clear operation is not supported by this collection

Reimplemented from **decaf::util::AbstractCollection< E >** (p.123).

6.646.3.5 `template<typename E> virtual bool decaf::util::StlList< E >::contains (
const E & value) const throw (lang::Exception) [inline, virtual]`

Returns true if this collection contains the specified element.

This implementation iterates over the elements in the collection, checking each element in turn for equality with the specified element.

Parameters

value - the value whose presence is to be queried for in this **Collection** (p.982).

Returns

true if the value is contained in this collection

Exceptions

Exception if an error occurs,

Reimplemented from **decaf::util::AbstractCollection< E >** (p.124).

6.646.3.6 `template<typename E> virtual void decaf::util::StlList< E >::copy (const StlList< E > & source) [inline, virtual]`

Referenced by `decaf::util::StlList< Pointer< BackupTransport > >::StlList()`.

6.646.3.7 `template<typename E> virtual bool decaf::util::StlList< E >::equals (const StlList< E > & source) const [inline, virtual]`

6.646.3.8 `template<typename E> virtual E decaf::util::StlList< E >::get (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException) [inline, virtual]`

Gets the element contained at position passed.

Parameters

index - position to get

Returns

value at index

Implements `decaf::util::List< E >` (p. 1868).

6.646.3.9 `template<typename E> virtual std::size_t decaf::util::StlList< E >::indexOf (const E & value) throw (decaf::lang::exceptions::NoSuchElementException) [inline, virtual]`

Returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element.

More formally, returns the lowest index *i* such that `get(i) == value`, or -1 if there is no such index.

Parameters

value - element to search for

Returns

the index of the first occurrence of the specified element in this list,

Exceptions

NoSuchElementException if value is not in the list

Implements `decaf::util::List< E >` (p. 1868).

6.646.3.10 `template<typename E> virtual bool decaf::util::StlList< E >::isEmpty () const [inline, virtual]`

Returns true if this collection contains no elements.

This implementation returns `size() (p. 990) == 0`.

Returns

true if the size method return 0.

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 125).

6.646.3.11 `template<typename E> virtual Iterator<E>* decaf::util::StlList< E >::iterator () [inline, virtual]`

6.646.3.12 `template<typename E> virtual Iterator<E>* decaf::util::StlList< E >::iterator () const [inline, virtual]`

6.646.3.13 `template<typename E> virtual std::size_t decaf::util::StlList< E >::lastIndexOf (const E & value) throw (decaf::lang::exceptions::NoSuchElementException) [inline, virtual]`

Returns the index of the last occurrence of the specified element in this list, or -1 if this list does not contain the element.

More formally, returns the highest index i such that get(i) == value or -1 if there is no such index.

Parameters

value - element to search for

Returns

the index of the last occurrence of the specified element in this list.

Exceptions

NoSuchElementException if value is not in the list

Implements **decaf::util::List< E >** (p. 1869).

6.646.3.14 `template<typename E> virtual ListIterator<E>* decaf::util::StlList< E >::listIterator (std::size_t index) throw (decaf::lang::exceptions::IndexOutOfBoundsException) [inline, virtual]`

Parameters

index index of first element to be returned from the list iterator (by a call to the next method).

Returns

a list iterator of the elements in this list (in proper sequence), starting at the specified position in this list. The specified index indicates the first element that would be returned by an initial call to next. An initial call to previous would return the element with the specified index minus one.

Exceptions

IndexOutOfBoundsException if the index is out of range (index < 0 || index > size() (p. 990))

Implements **decaf::util::List< E >** (p. 1870).

6.646.3.15 `template<typename E> virtual ListIterator<E>* decaf::util::StlList< E >::listIterator () [inline, virtual]`

Returns

a list iterator over the elements in this list (in proper sequence).

Implements `decaf::util::List< E >` (p.1869).

6.646.3.16 `template<typename E> virtual ListIterator<E>* decaf::util::StlList< E >::listIterator () const [inline, virtual]`

Implements `decaf::util::List< E >` (p.1869).

6.646.3.17 `template<typename E> virtual ListIterator<E>* decaf::util::StlList< E >::listIterator (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException) [inline, virtual]`

Implements `decaf::util::List< E >` (p.1869).

6.646.3.18 `template<typename E> virtual E decaf::util::StlList< E >::remove (std::size_t index) throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IndexOutOfBoundsException) [inline, virtual]`

Removes the element at the specified position in this list.

Shifts any subsequent elements to the left (subtracts one from their indices). Returns the element that was removed from the list.

Parameters

index - the index of the element to be removed

Returns

the element previously at the specified position

Exceptions

IndexOutOfBoundsException - if the index is greater than size

UnsupportedOperationException - If the collection is non-modifiable.

Implements `decaf::util::List< E >` (p.1870).

6.646.3.19 `template<typename E> virtual bool decaf::util::StlList< E >::remove (const E & value) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException) [inline, virtual]`

Removes a single instance of the specified element from this collection, if it is present (optional operation).

More formally, removes the first element *e* such that *e* == *o*, if this collection contains one or more such elements. Returns true if this collection contained the specified element (or equivalently, if this collection changed as a result of the call).

This implementation iterates over the collection looking for the specified element. If it finds the element, it removes the element from the collection using the iterator's remove method.

Note that this implementation throws an `UnsupportedOperationException` if the iterator returned by this collection's iterator method does not implement the remove method and this collection contains the specified object.

Parameters

value - element to be removed from this collection, if present

Returns

true if an element was removed as a result of this call

Exceptions

UnsupportedOperationException if the remove operation is not supported by this collection.

IllegalArgumentException If the value is not a valid entry for this **Collection** (p. 982).

Reimplemented from `decaf::util::AbstractCollection< E >` (p. 127).

```
6.646.3.20  template<typename E> virtual E decaf::util::StlList< E
            >::set ( std::size_t index, const E & element ) throw (
            decaf::lang::exceptions::IndexOutOfBoundsException ) [inline,
            virtual]
```

Replaces the element at the specified position in this list with the specified element.

Parameters

index - index of the element to replace

element - element to be stored at the specified position

Returns

the element previously at the specified position

Exceptions

IndexOutOfBoundsException - if the index is greater than size

Implements `decaf::util::List< E >` (p. 1871).

```
6.646.3.21  template<typename E> virtual std::size_t decaf::util::StlList< E >::size
            ( ) const [inline, virtual]
```

Returns the number of elements in this collection.

If this collection contains more than `Integer.MAX_VALUE` elements, returns `Integer.MAX_VALUE`.

Returns

the number of elements in this collection

Implements **decaf::util::Collection**< **E** > (p. 990).

Referenced by `decaf::util::StlList< Pointer< BackupTransport > >::add()`, `decaf::util::StlList< Pointer< BackupTransport > >::addAll()`, `decaf::util::StlList< Pointer< BackupTransport > >::get()`, `decaf::util::StlList< Pointer< BackupTransport > >::lastIndexOf()`, `decaf::util::StlList< Pointer< BackupTransport > >::listIterator()`, `decaf::util::StlList< Pointer< BackupTransport > >::remove()`, and `decaf::util::StlList< Pointer< BackupTransport > >::set()`.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/StlList.h`

6.647 **decaf::util::StlMap**< **K**, **V**, **COMPARATOR** > Class Template Reference

Map (p. 1970) template that wraps around a `std::map` to provide a more user-friendly interface and to provide common functions that do not exist in `std::map`.

```
#include <src/main/decaf/util/StlMap.h>
```

Inheritance diagram for `decaf::util::StlMap`< **K**, **V**, **COMPARATOR** >:

Public Member Functions

- **StlMap** ()
Default constructor - does nothing.
- **StlMap** (const **StlMap** &source)
Copy constructor - copies the content of the given map into this one.
- **StlMap** (const **Map**< **K**, **V**, **COMPARATOR** > &source)
Copy constructor - copies the content of the given map into this one.
- virtual **~StlMap** ()
- virtual bool **equals** (const **StlMap** &source) const
- virtual bool **equals** (const **Map**< **K**, **V**, **COMPARATOR** > &source) const
Comparison, equality is dependent on the method of determining if the element are equal.
Parameters
source - **Map** (p. 1970) to compare to this one.
Returns
true if the Map (p. 1970) passed is equal in value to this one.
- virtual void **copy** (const **StlMap** &source)

- virtual void **copy** (const **Map**< K, V, COMPARATOR > &source)
*Copies the content of the source map into this map.
Erases all existing data in this map.*
Parameters
source The source object to copy from.
- virtual void **clear** () throw (decaf::lang::exceptions::UnsupportedOperationException)
Removes all keys and values from this map.
Exceptions
UnsupportedOperationException if this map is unmodifiable.
- virtual bool **containsKey** (const K &key) const
Indicates whether or this map contains a value for the given key.
Parameters
key The key to look up.
Returns
true if this map contains the value, otherwise false.
- virtual bool **containsValue** (const V &value) const
Indicates whether or this map contains a value for the given value, i.e. they are equal, this is done by operator== so the types must pass equivalence testing in this manner.
Parameters
value The Value to look up.
Returns
true if this map contains the value, otherwise false.
- virtual bool **isEmpty** () const
Returns
*if the **Map** (p. 1970) contains any element or not, TRUE or FALSE*
- virtual std::size_t **size** () const
Returns
The number of elements (key/value pairs) in this map.
- virtual V & **get** (const K &key) throw (lang::exceptions::NoSuchElementException)
*Gets the value mapped to the specified key in the **Map** (p. 1970).
If there is no element in the map whose key is equivalent to the key provided then a NoSuchElementException is thrown.*
Parameters
key The search key.
Returns
A reference to the value for the given key.
Exceptions
NoSuchElementException if the key requests doesn't exist in the **Map** (p. 1970).

- virtual const V & **get** (const K &key) const throw (decaf::lang::exceptions::NoSuchElementException)

*Gets the value mapped to the specified key in the **Map** (p. 1970).*

*If there is no element in the map whose key is equivalent to the key provided then a *NoSuchElementException* is thrown.*

Parameters

key The search key.

Returns

A {const} reference to the value for the given key.

Exceptions

NoSuchElementException if the key requests doesn't exist in the **Map** (p. 1970).

- virtual void **put** (const K &key, const V &value) throw (decaf::lang::exceptions::UnsupportedOperationException)

Sets the value for the specified key.

Parameters

key The target key.

value The value to be set.

Exceptions

UnsupportedOperationException if this map is unmodifiable.

- virtual void **putAll** (const **StlMap**< K, V, COMPARATOR > &other) throw (decaf::lang::exceptions::UnsupportedOperationException)

- virtual void **putAll** (const **Map**< K, V, COMPARATOR > &other) throw (decaf::lang::exceptions::UnsupportedOperationException)

*Stores a copy of the Mappings contained in the other **Map** (p. 1970) in this one.*

Parameters

other A **Map** (p. 1970) instance whose elements are to all be inserted in this **Map** (p. 1970).

Exceptions

UnsupportedOperationException If the implementing class does not support the putAll operation.

- virtual V **remove** (const K &key) throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException)

Removes the value (key/value pair) for the specified key from the map, returns a copy of the value that was mapped to the key.

Parameters

key The search key.

Returns

a copy of the element that was previously mapped to the given key

Exceptions

NoSuchElementException if this key is not in the **Map** (p. 1970).
UnsupportedOperationException if this map is unmodifiable.

- virtual std::vector< K > **keySet** () const

Returns a **Set** (p. 2729) view of the mappings contained in this map.

The set is backed by the map, so changes to the map are reflected in the set, and vice-versa. If the map is modified while an iteration over the set is in progress (except through the iterator's own remove operation, or through the setValue operation on a map entry returned by the iterator) the results of the iteration are undefined. The set supports element removal, which removes the corresponding mapping from the map, via the **Iterator.remove** (p. 1718), **Set.remove** (p. 127), removeAll, retainAll and clear operations. It does not support the add or addAll operations.

Returns

the entire set of keys in this map as a std::vector.

- virtual std::vector< V > **values** () const

Returns

the entire set of values in this map as a std::vector.

- virtual void **lock** () throw (decaf::lang::exceptions::RuntimeException)

Locks the object.

- virtual bool **tryLock** () throw (decaf::lang::exceptions::RuntimeException)

Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

- virtual void **unlock** () throw (decaf::lang::exceptions::RuntimeException)

Unlocks the object.

- virtual void **wait** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)

Waits on a signal from this object, which is generated by a call to Notify.

- virtual void **wait** (long long millisecs) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)

Waits on a signal from this object, which is generated by a call to Notify.

- virtual void **wait** (long long millisecs, int nanos) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)

Waits on a signal from this object, which is generated by a call to Notify.

- virtual void **notify** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)

Signals a waiter on this object that it can now wake up and continue.

- virtual void **notifyAll** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)

Signals the waiters on this object that it can now wake up and continue.

6.647.1 Detailed Description

```
template<typename K, typename V, typename COMPARATOR = std::less<K>>
class decaf::util::StlMap< K, V, COMPARATOR >
```

Map (p.1970) template that wraps around a `std::map` to provide a more user-friendly interface and to provide common functions that do not exist in `std::map`.

Since

1.0

6.647.2 Constructor & Destructor Documentation

6.647.2.1 `template<typename K, typename V, typename COMPARATOR = std::less<K>> decaf::util::StlMap< K, V, COMPARATOR >::StlMap () [inline]`

Default constructor - does nothing.

6.647.2.2 `template<typename K, typename V, typename COMPARATOR = std::less<K>> decaf::util::StlMap< K, V, COMPARATOR >::StlMap (const StlMap< K, V, COMPARATOR > & source) [inline]`

Copy constructor - copies the content of the given map into this one.

Parameters

source The source map.

6.647.2.3 `template<typename K, typename V, typename COMPARATOR = std::less<K>> decaf::util::StlMap< K, V, COMPARATOR >::StlMap (const Map< K, V, COMPARATOR > & source) [inline]`

Copy constructor - copies the content of the given map into this one.

Parameters

source The source map.

6.647.2.4 `template<typename K, typename V, typename COMPARATOR =
std::less<K>> virtual decaf::util::StlMap< K, V, COMPARATOR
>::~StlMap () [inline, virtual]`

6.647.3 Member Function Documentation

6.647.3.1 `template<typename K, typename V, typename COMPARATOR
= std::less<K>> virtual void decaf::util::StlMap<
K, V, COMPARATOR >::clear () throw (
decaf::lang::exceptions::UnsupportedOperationException) [inline,
virtual]`

Removes all keys and values from this map.

Exceptions

UnsupportedOperationException if this map is unmodifiable.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p.1972).

Referenced by `decaf::util::StlMap< std::string, cms::Topic * >::copy()`.

6.647.3.2 `template<typename K, typename V, typename COMPARATOR =
std::less<K>> virtual bool decaf::util::StlMap< K, V, COMPARATOR
>::containsKey (const K & key) const [inline, virtual]`

Indicates whether or this map contains a value for the given key.

Parameters

key The key to look up.

Returns

true if this map contains the value, otherwise false.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p.1973).

Referenced by `decaf::util::StlMap< std::string, cms::Topic * >::equals()`.

6.647.3.3 `template<typename K, typename V, typename COMPARATOR =
std::less<K>> virtual bool decaf::util::StlMap< K, V, COMPARATOR
>::containsValue (const V & value) const [inline, virtual]`

Indicates whether or this map contains a value for the given value, i.e.

they are equal, this is done by operator== so the types must pass equivalence testing in this manner.

Parameters

value The Value to look up.

Returns

true if this map contains the value, otherwise false.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p.1973).

6.647.3.4 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::StlMap< K, V, COMPARATOR >::copy (const StlMap< K, V, COMPARATOR > & source) [inline, virtual]`

Referenced by `decaf::util::StlMap< std::string, cms::Topic * >::StlMap()`.

6.647.3.5 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::StlMap< K, V, COMPARATOR >::copy (const Map< K, V, COMPARATOR > & source) [inline, virtual]`

Copies the content of the source map into this map.

Erases all existing data in this map.

Parameters

source The source object to copy from.

Implements `decaf::util::Map< K, V, COMPARATOR >` (p. 1974).

6.647.3.6 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual bool decaf::util::StlMap< K, V, COMPARATOR >::equals (const Map< K, V, COMPARATOR > & source) const [inline, virtual]`

Comparison, equality is dependent on the method of determining if the element are equal.

Parameters

source - `Map` (p. 1970) to compare to this one.

Returns

true if the `Map` (p. 1970) passed is equal in value to this one.

Implements `decaf::util::Map< K, V, COMPARATOR >` (p. 1974).

6.647.3.7 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual bool decaf::util::StlMap< K, V, COMPARATOR >::equals (const StlMap< K, V, COMPARATOR > & source) const [inline, virtual]`

6.647.3.8 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual V& decaf::util::StlMap< K, V, COMPARATOR >::get (const K & key) throw (lang::exceptions::NoSuchElementException) [inline, virtual]`

Gets the value mapped to the specified key in the `Map` (p. 1970).

If there is no element in the map whose key is equivalent to the key provided then a `NoSuchElementException` is thrown.

Parameters

key The search key.

Returns

A reference to the value for the given key.

Exceptions

NoSuchElementException if the key requests doesn't exist in the **Map** (p. 1970).

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 1975).

```
6.647.3.9  template<typename K, typename V, typename COMPARATOR
           = std::less<K>> virtual const V& decaf::util::StlMap< K, V,
           COMPARATOR >::get ( const K & key ) const throw (
           lang::exceptions::NoSuchElementException ) [inline, virtual]
```

Gets the value mapped to the specified key in the **Map** (p. 1970).

If there is no element in the map whose key is equivalent to the key provided then a *NoSuchElementException* is thrown.

Parameters

key The search key.

Returns

A {const} reference to the value for the given key.

Exceptions

NoSuchElementException if the key requests doesn't exist in the **Map** (p. 1970).

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 1976).

```
6.647.3.10 template<typename K, typename V, typename COMPARATOR =
           std::less<K>> virtual bool decaf::util::StlMap< K, V, COMPARATOR
           >::isEmpty ( ) const [inline, virtual]
```

Returns

if the **Map** (p. 1970) contains any element or not, TRUE or FALSE

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 1977).

```
6.647.3.11 template<typename K, typename V, typename COMPARATOR =
           std::less<K>> virtual std::vector<K> decaf::util::StlMap< K, V,
           COMPARATOR >::keySet ( ) const [inline, virtual]
```

Returns a **Set** (p. 2729) view of the mappings contained in this map.

The set is backed by the map, so changes to the map are reflected in the set, and vice-versa. If the map is modified while an iteration over the set is in progress (except through the iterator's own

remove operation, or through the setValue operation on a map entry returned by the iterator) the results of the iteration are undefined. The set supports element removal, which removes the corresponding mapping from the map, via the **Iterator.remove** (p. 1718), **Set.remove** (p. 127), removeAll, retainAll and clear operations. It does not support the add or addAll operations.

Returns

the entire set of keys in this map as a std::vector.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 1977).

6.647.3.12 `template<typename K, typename V, typename COMPARATOR =
std::less<K>> virtual void decaf::util::StlMap< K, V, COMPARATOR
>::lock () throw (decaf::lang::exceptions::RuntimeException)
[inline, virtual]`

Locks the object.

Exceptions

RuntimeException if an error occurs while locking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2932).

6.647.3.13 `template<typename K, typename V, typename COMPARATOR =
std::less<K>> virtual void decaf::util::StlMap< K, V, COMPARATOR
>::notify () throw (decaf::lang::exceptions::RuntimeException,
decaf::lang::exceptions::IllegalMonitorStateException) [inline,
virtual]`

Signals a waiter on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

RuntimeException if an error occurs while notifying one of the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 2933).

6.647.3.14 `template<typename K, typename V, typename COMPARATOR =
std::less<K>> virtual void decaf::util::StlMap< K, V, COMPARATOR
>::notifyAll () throw (decaf::lang::exceptions::RuntimeException,
decaf::lang::exceptions::IllegalMonitorStateException) [inline,
virtual]`

Signals the waiters on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

RuntimeException if an error occurs while notifying the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 2934).

6.647.3.15 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::StlMap< K, V, COMPARATOR >::put (const K & key, const V & value) throw (decaf::lang::exceptions::UnsupportedOperationException) [inline, virtual]`

Sets the value for the specified key.

Parameters

key The target key.

value The value to be set.

Exceptions

UnsupportedOperationException if this map is unmodifiable.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 1978).

Referenced by **decaf::util::StlMap< std::string, cms::Topic * >::putAll()**.

6.647.3.16 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::StlMap< K, V, COMPARATOR >::putAll (const StlMap< K, V, COMPARATOR > & other) throw (decaf::lang::exceptions::UnsupportedOperationException) [inline, virtual]`

Referenced by **decaf::util::StlMap< std::string, cms::Topic * >::copy()**.

6.647.3.17 `template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::StlMap< K, V, COMPARATOR >::putAll (const Map< K, V, COMPARATOR > & other) throw (decaf::lang::exceptions::UnsupportedOperationException) [inline, virtual]`

Stores a copy of the Mappings contained in the other **Map** (p. 1970) in this one.

Parameters

other A **Map** (p. 1970) instance whose elements are to all be inserted in this **Map** (p. 1970).

Exceptions

UnsupportedOperationException If the implementing class does not support the putAll operation.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 1979).

6.647.3.18 `template<typename K, typename V, typename COMPARATOR
= std::less<K>> virtual V decaf::util::StlMap< K, V,
COMPARATOR >::remove (const K & key) throw
(decaf::lang::exceptions::NoSuchElementException,
decaf::lang::exceptions::UnsupportedOperationException) [inline,
virtual]`

Removes the value (key/value pair) for the specified key from the map, returns a copy of the value that was mapped to the key.

Parameters

key The search key.

Returns

a copy of the element that was previously mapped to the given key

Exceptions

NoSuchElementException if this key is not in the **Map** (p. 1970).

UnsupportedOperationException if this map is unmodifiable.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 1980).

6.647.3.19 `template<typename K, typename V, typename COMPARATOR
= std::less<K>> virtual std::size_t decaf::util::StlMap< K, V,
COMPARATOR >::size () const [inline, virtual]`

Returns

The number of elements (key/value pairs) in this map.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 1981).

6.647.3.20 `template<typename K, typename V, typename COMPARATOR =
std::less<K>> virtual bool decaf::util::StlMap< K, V, COMPARATOR
>::tryLock () throw (decaf::lang::exceptions::RuntimeException)
[inline, virtual]`

Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

Returns

true if the lock was acquired, false if it is already held by another thread.

Exceptions

RuntimeException if an error occurs while locking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2935).

6.647.3.21 `template<typename K, typename V, typename COMPARATOR =
std::less<K>> virtual void decaf::util::StlMap< K, V, COMPARATOR
>::unlock () throw (decaf::lang::exceptions::RuntimeException)
[inline, virtual]`

Unlocks the object.

Exceptions

RuntimeException if an error occurs while unlocking the object.

Implements `decaf::util::concurrent::Synchronizable` (p. 2936).

6.647.3.22 `template<typename K, typename V, typename COMPARATOR =
std::less<K>> virtual std::vector<V> decaf::util::StlMap< K, V,
COMPARATOR >::values () const [inline, virtual]`

Returns

the entire set of values in this map as a `std::vector`.

Implements `decaf::util::Map< K, V, COMPARATOR >` (p. 1981).

Referenced by `decaf::util::StlMap< std::string, cms::Topic * >::values()`.

6.647.3.23 `template<typename K, typename V, typename COMPARATOR =
std::less<K>> virtual void decaf::util::StlMap< K, V, COMPARATOR
>::wait () throw (decaf::lang::exceptions::RuntimeException,
decaf::lang::exceptions::IllegalMonitorStateException,
decaf::lang::exceptions::InterruptedException) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to `Notify`.

Must have this object locked before calling.

Exceptions

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements `decaf::util::concurrent::Synchronizable` (p. 2937).

6.647.3.24 `template<typename K, typename V, typename COMPARATOR
= std::less<K>> virtual void decaf::util::StlMap< K, V,
COMPARATOR >::wait (long long millisecs, int
nanos) throw (decaf::lang::exceptions::RuntimeException,
decaf::lang::exceptions::IllegalArgumentException,
decaf::lang::exceptions::IllegalMonitorStateException,
decaf::lang::exceptions::InterruptedException) [inline, virtual]`

Waits on a signal from this object, which is generated by a call to `Notify`.

Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

Parameters

milliseconds the time in milliseconds to wait, or WAIT_INFINITE

nanos additional time in nanoseconds with a range of 0-999999

Exceptions

IllegalArgumentException if an error occurs or the nanos argument is not in the range of [0-999999]

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2940).

```
6.647.3.25  template<typename K, typename V, typename COMPARATOR
            = std::less<K>> virtual void decaf::util::StlMap< K,
            V, COMPARATOR >::wait ( long long  milliseconds )
            throw ( decaf::lang::exceptions::RuntimeException,
            decaf::lang::exceptions::IllegalMonitorStateException,
            decaf::lang::exceptions::InterruptedException ) [inline, virtual]
```

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval.

Parameters

milliseconds the time in milliseconds to wait, or WAIT_INFINITE

Exceptions

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2938).

The documentation for this class was generated from the following file:

- src/main/decaf/util/StlMap.h

6.648 decaf::util::StlQueue< T > Class Template Reference

The **Queue** (p. 2507) class accepts messages with an `psuh(m)` command where `m` is the message to be queued.

```
#include <src/main/decaf/util/StlQueue.h>
```

Inheritance diagram for `decaf::util::StlQueue< T >`:

Data Structures

- class **QueueIterator**

Public Member Functions

- **StlQueue** ()
- virtual **~StlQueue** ()
- **Iterator**< T > * **iterator** ()
*Gets an **Iterator** (p. 1716) over this **Queue** (p. 2507).*
- void **clear** ()
Empties this queue.
- T & **front** ()
Returns a Reference to the element at the head of the queue.
- const T & **front** () const
Returns a Reference to the element at the head of the queue.
- T & **back** ()
Returns a Reference to the element at the tail of the queue.
- const T & **back** () const
Returns a Reference to the element at the tail of the queue.
- void **push** (const T &t)
Places a new Object at the Tail of the queue.
- void **enqueueFront** (const T &t)
Places a new Object at the front of the queue.
- T **pop** ()
Removes and returns the element that is at the Head of the queue.
- size_t **size** () const
*Gets the Number of elements currently in the **Queue** (p. 2507).*
- bool **empty** () const

*Checks if this **Queue** (p. 2507) is currently empty.*

- virtual std::vector< T > **toArray** () const
- void **reverse** (**StlQueue**< T > &target) const
Reverses the order of the contents of this queue and stores them in the target queue.
- virtual void **lock** () throw (decaf::lang::exceptions::RuntimeException)
Locks the object.
- virtual bool **tryLock** () throw (decaf::lang::exceptions::RuntimeException)
Attempts to Lock the object, if the lock is already held by another thread than this method returns false.
- virtual void **unlock** () throw (decaf::lang::exceptions::RuntimeException)
Unlocks the object.
- virtual void **wait** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (long long millisecs) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (long long millisecs, int nanos) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **notify** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)
Signals a waiter on this object that it can now wake up and continue.
- virtual void **notifyAll** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)
Signals the waiters on this object that it can now wake up and continue.
- T & **getSafeValue** ()
Fetch a reference to the safe value this object will return when there is nothing to fetch from the queue.

6.648.1 Detailed Description

```
template<typename T> class decaf::util::StlQueue< T >
```

The **Queue** (p. 2507) class accepts messages with an psuh(m) command where m is the message to be queued. It destructively returns the message with **pop()** (p. 2863). **pop()** (p. 2863) returns messages in the order they were enqueued.

Queue (p. 2507) is implemented with an instance of the STL queue object. The interface is essentially the same as that of the STL queue except that the pop method actually reaturns a reference to the element popped. This frees the app from having to call the **front** method before calling pop.

```
Queue<string> sq; // make a queue to hold string messages sq.push(s); // enqueues a message
m string s = sq.pop(); // dequeues a message
```

= DESIGN CONSIDERATIONS

The **Queue** (p. 2507) class inherits from the Synchronizable interface and provides methods for locking and unlocking this queue as well as waiting on this queue. In a multi-threaded app this can allow for multiple threads to be reading from and writing to the same **Queue** (p. 2507).

Clients should consider that in a multiple threaded app it is possible that items could be placed on the queue faster than you are taking them off, so protection should be placed in your polling loop to ensure that you don't get stuck there.

6.648.2 Constructor & Destructor Documentation

6.648.2.1 `template<typename T> decaf::util::StlQueue< T >::StlQueue ()`
[inline]

6.648.2.2 `template<typename T> virtual decaf::util::StlQueue< T >::~StlQueue ()`
[inline, virtual]

6.648.3 Member Function Documentation

6.648.3.1 `template<typename T> T& decaf::util::StlQueue< T >::back ()`
[inline]

Returns a Reference to the element at the tail of the queue.

Returns

reference to a queue type object or (safe)

6.648.3.2 `template<typename T> const T& decaf::util::StlQueue< T >::back ()`
const [inline]

Returns a Reference to the element at the tail of the queue.

Returns

reference to a queue type object or (safe)

6.648.3.3 `template<typename T> void decaf::util::StlQueue< T >::clear ()`
[inline]

Empties this queue.

6.648.3.4 `template<typename T> bool decaf::util::StlQueue< T >::empty ()
 const [inline]`

Checks if this **Queue** (p. 2507) is currently empty.

Returns

boolean indicating queue emptiness

6.648.3.5 `template<typename T> void decaf::util::StlQueue< T >::enqueueFront ()
 const T & t) [inline]`

Places a new Object at the front of the queue.

Parameters

t - **Queue** (p. 2507) Object Type reference.

6.648.3.6 `template<typename T> const T& decaf::util::StlQueue< T >::front ()
 const [inline]`

Returns a Reference to the element at the head of the queue.

Returns

reference to a queue type object or (safe)

6.648.3.7 `template<typename T> T& decaf::util::StlQueue< T >::front ()
 [inline]`

Returns a Reference to the element at the head of the queue.

Returns

reference to a queue type object or (safe)

6.648.3.8 `template<typename T> T& decaf::util::StlQueue< T >::getSafeValue ()
 [inline]`

Fetch a reference to the safe value this object will return when there is nothing to fetch from the queue.

Returns

Reference to this Queues safe object

Referenced by `decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >::back()`, `decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >::front()`, and `decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >::pop()`.

6.648.3.9 `template<typename T> Iterator<T>* decaf::util::StlQueue< T >::iterator () [inline]`

Gets an **Iterator** (p. 1716) over this **Queue** (p. 2507).

Returns

new iterator pointer that is owned by the caller.

6.648.3.10 `template<typename T> virtual void decaf::util::StlQueue< T >::lock () throw (decaf::lang::exceptions::RuntimeException) [inline, virtual]`

Locks the object.

Exceptions

RuntimeException if an error occurs while locking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2932).

6.648.3.11 `template<typename T> virtual void decaf::util::StlQueue< T >::notify () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException) [inline, virtual]`

Signals a waiter on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

RuntimeException if an error occurs while notifying one of the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 2933).

6.648.3.12 `template<typename T> virtual void decaf::util::StlQueue< T >::notifyAll () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException) [inline, virtual]`

Signals the waiters on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

RuntimeException if an error occurs while notifying the waiting threads.

Implements **decaf::util::concurrent::Synchronizable** (p. 2934).

6.648.3.13 `template<typename T> T decaf::util::StlQueue< T >::pop ()`
[inline]

Removes and returns the element that is at the Head of the queue.

Returns

reference to a queue type object or (safe)

6.648.3.14 `template<typename T> void decaf::util::StlQueue< T >::push (const T & t)` [inline]

Places a new Object at the Tail of the queue.

Parameters

t - **Queue** (p. 2507) Object Type reference.

6.648.3.15 `template<typename T> void decaf::util::StlQueue< T >::reverse (StlQueue< T > & target)` const [inline]

Reverses the order of the contents of this queue and stores them in the target queue.

Parameters

target - The target queue that will receive the contents of this queue in reverse order.

6.648.3.16 `template<typename T> size_t decaf::util::StlQueue< T >::size ()` const [inline]

Gets the Number of elements currently in the **Queue** (p. 2507).

Returns

Queue (p. 2507) Size

6.648.3.17 `template<typename T> virtual std::vector<T> decaf::util::StlQueue< T >::toArray ()` const [inline, virtual]

Returns

the all values in this queue as a std::vector.

6.648.3.18 `template<typename T> virtual bool decaf::util::StlQueue< T >::tryLock ()` throw (decaf::lang::exceptions::RuntimeException) [inline, virtual]

Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

Returns

true if the lock was acquired, false if it is already held by another thread.

Exceptions

RuntimeException if an error occurs while locking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2935).

```
6.648.3.19  template<typename T> virtual void decaf::util::StlQueue< T >::unlock  
            ( ) throw ( decaf::lang::exceptions::RuntimeException ) [inline,  
            virtual]
```

Unlocks the object.

Exceptions

RuntimeException if an error occurs while unlocking the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2936).

```
6.648.3.20  template<typename T> virtual void decaf::util::StlQueue< T  
            >::wait ( ) throw ( decaf::lang::exceptions::RuntimeException,  
            decaf::lang::exceptions::IllegalMonitorStateException,  
            decaf::lang::exceptions::InterruptedException ) [inline, virtual]
```

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling.

Exceptions

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2937).

```
6.648.3.21  template<typename T> virtual void decaf::util::StlQueue< T >::wait (  
            long long millisecs ) throw ( decaf::lang::exceptions::RuntimeException,  
            decaf::lang::exceptions::IllegalMonitorStateException,  
            decaf::lang::exceptions::InterruptedException ) [inline, virtual]
```

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval.

Parameters

millisecs the time in milliseconds to wait, or WAIT_INFINITE

Exceptions

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2938).

```
6.648.3.22  template<typename T> virtual void decaf::util::StlQueue<
              T >::wait ( long long millisecs, int nanos )
              throw ( decaf::lang::exceptions::RuntimeException,
                      decaf::lang::exceptions::IllegalArgumentException,
                      decaf::lang::exceptions::IllegalMonitorStateException,
                      decaf::lang::exceptions::InterruptedException ) [inline, virtual]
```

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

Parameters

millisecs the time in milliseconds to wait, or WAIT_INFINITE

nanos additional time in nanoseconds with a range of 0-999999

Exceptions

IllegalArgumentException if an error occurs or the nanos argument is not in the range of [0-999999]

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2940).

The documentation for this class was generated from the following file:

- src/main/decaf/util/StlQueue.h

6.649 decaf::util::StlSet< E > Class Template Reference

Set (p. 2729) template that wraps around a std::set to provide a more user-friendly interface and to provide common functions that do not exist in std::set.

```
#include <src/main/decaf/util/StlSet.h>
```

Inheritance diagram for decaf::util::StlSet< E >:

Data Structures

- class **ConstSetIterator**
- class **SetIterator**

Public Member Functions

- **StlSet** ()
Default constructor - does nothing.
- **StlSet** (const **StlSet** &source)
Copy constructor - copies the content of the given set into this one.
- **StlSet** (const **Collection**< E > &source)
Copy constructor - copies the content of the given set into this one.
- virtual ~**StlSet** ()
- **Iterator**< E > * **iterator** ()
- **Iterator**< E > * **iterator** () const
- virtual bool **equals** (const **StlSet** &source) const
- virtual void **copy** (const **StlSet** &source)
- virtual void **clear** () throw (lang::exceptions::UnsupportedOperationException)
*Removes all of the elements from this collection (optional operation).
The collection will be empty after this method returns.
This implementation iterates over this collection, removing each element using the **Iterator.remove** (p. 1718) operation. Most implementations will probably choose to override this method for efficiency.
Note that this implementation will throw an **UnsupportedOperationException** if the iterator returned by this collection's iterator method does not implement the remove method and this collection is non-empty.*
Exceptions
***UnsupportedOperationException** if the clear operation is not supported by this collection*
- virtual bool **contains** (const E &value) const throw (lang::Exception)
*Returns true if this collection contains the specified element.
This implementation iterates over the elements in the collection, checking each element in turn for equality with the specified element.*
Parameters
*value - the value whose presence is to be queried for in this **Collection** (p. 982).*
Returns
true if the value is contained in this collection
Exceptions
***Exception** if an error occurs,*

- virtual bool **isEmpty** () const
- virtual std::size_t **size** () const
- virtual bool **add** (const E &value) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException)

Returns true if this collection changed as a result of the call.

(Returns false if this collection does not permit duplicates and already contains the specified element.)

*Collections that support this operation may place limitations on what elements may be added to this collection. In particular, some collections will refuse to add null elements, and others will impose restrictions on the type of elements that may be added. **Collection** (p. 982) classes should clearly specify in their documentation any restrictions on what elements may be added.*

If a collection refuses to add a particular element for any reason other than that it already contains the element, it must throw an exception (rather than returning false). This preserves the invariant that a collection always contains the specified element after this call returns.

For non-pointer values, i.e. class instances or string's the object will be copied into the collection, thus the object must support being copied, must not hide the copy constructor and assignment operator.

Parameters

value - reference to the element to add.

Returns

true if the element was added

Exceptions

UnsupportedOperationException

IllegalArgumentException

IllegalStateException if the element cannot be added at this time due to insertion restrictions

- virtual bool **remove** (const E &value) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException)

Removes a single instance of the specified element from this collection, if it is present (optional operation).

*More formally, removes the first element *e* such that *e* == *o*, if this collection contains one or more such elements. Returns true if this collection contained the specified element (or equivalently, if this collection changed as a result of the call).*

This implementation iterates over the collection looking for the specified element. If it finds the element, it removes the element from the collection using the iterator's remove method.

Note that this implementation throws an UnsupportedOperationException if the iterator returned by this collection's iterator method does not implement the remove method and this collection contains the specified object.

Parameters

value - element to be removed from this collection, if present

Returns

true if an element was removed as a result of this call

Exceptions

UnsupportedOperationException if the remove operation is not supported by this collection.

IllegalArgumentException If the value is not a valid entry for this **Collection** (p. 982).

6.649.1 Detailed Description

template<typename E> class decaf::util::StlSet< E >

Set (p. 2729) template that wraps around a `std::set` to provide a more user-friendly interface and to provide common functions that do not exist in `std::set`.

6.649.2 Constructor & Destructor Documentation

6.649.2.1 template<typename E> decaf::util::StlSet< E >::StlSet () [inline]

Default constructor - does nothing.

6.649.2.2 template<typename E> decaf::util::StlSet< E >::StlSet (const StlSet< E > & *source*) [inline]

Copy constructor - copies the content of the given set into this one.

Parameters

source The source set.

6.649.2.3 template<typename E> decaf::util::StlSet< E >::StlSet (const Collection< E > & *source*) [inline]

Copy constructor - copies the content of the given set into this one.

Parameters

source The source set.

6.649.2.4 template<typename E> virtual decaf::util::StlSet< E >::~~StlSet () [inline, virtual]

6.649.3 Member Function Documentation

6.649.3.1 template<typename E> virtual bool decaf::util::StlSet< E >::add (const E & *value*) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException) [inline, virtual]

Returns true if this collection changed as a result of the call.

(Returns false if this collection does not permit duplicates and already contains the specified element.)

Collections that support this operation may place limitations on what elements may be added to this collection. In particular, some collections will refuse to add null elements, and others will impose restrictions on the type of elements that may be added. **Collection** (p. 982) classes should clearly specify in their documentation any restrictions on what elements may be added.

If a collection refuses to add a particular element for any reason other than that it already contains the element, it must throw an exception (rather than returning false). This preserves the invariant that a collection always contains the specified element after this call returns.

For non-pointer values, i.e. class instances or string's the object will be copied into the collection, thus the object must support being copied, must not hide the copy constructor and assignment operator.

Parameters

value - reference to the element to add.

Returns

true if the element was added

Exceptions

UnsupportedOperationException

IllegalArgumentException

IllegalStateException if the element cannot be added at this time due to insertion restrictions

Implements **decaf::util::Collection< E >** (p. 984).

6.649.3.2 `template<typename E> virtual void decaf::util::StlSet< E >::clear ()
throw (lang::exceptions::UnsupportedOperationException) [inline,
virtual]`

Removes all of the elements from this collection (optional operation).

The collection will be empty after this method returns.

This implementation iterates over this collection, removing each element using the **Iterator.remove** (p.1718) operation. Most implementations will probably choose to override this method for efficiency.

Note that this implementation will throw an `UnsupportedOperationException` if the iterator returned by this collection's iterator method does not implement the remove method and this collection is non-empty.

Exceptions

UnsupportedOperationException if the clear operation is not supported by this collection

Reimplemented from **decaf::util::AbstractCollection< E >** (p.123).

6.649.3.3 `template<typename E> virtual bool decaf::util::StlSet< E >::contains (
const E & value) const throw (lang::Exception) [inline, virtual]`

Returns true if this collection contains the specified element.

This implementation iterates over the elements in the collection, checking each element in turn for equality with the specified element.

Parameters

value - the value whose presence is to be queried for in this **Collection** (p. 982).

Returns

true if the value is contained in this collection

Exceptions

Exception if an error occurs,

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 124).

6.649.3.4 `template<typename E> virtual void decaf::util::StlSet< E >::copy (const StlSet< E > & source) [inline, virtual]`

Referenced by `decaf::util::StlSet< ActiveMQSession * >::StlSet()`.

6.649.3.5 `template<typename E> virtual bool decaf::util::StlSet< E >::equals (const StlSet< E > & source) const [inline, virtual]`

6.649.3.6 `template<typename E> virtual bool decaf::util::StlSet< E >::isEmpty () const [inline, virtual]`

Returns

if the set contains any element or not, TRUE or FALSE

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 125).

6.649.3.7 `template<typename E> Iterator<E>* decaf::util::StlSet< E >::iterator () const [inline]`

6.649.3.8 `template<typename E> Iterator<E>* decaf::util::StlSet< E >::iterator () [inline]`

6.649.3.9 `template<typename E> virtual bool decaf::util::StlSet< E >::remove (const E & value) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException) [inline, virtual]`

Removes a single instance of the specified element from this collection, if it is present (optional operation).

More formally, removes the first element *e* such that *e* == *o*, if this collection contains one or more such elements. Returns true if this collection contained the specified element (or equivalently, if this collection changed as a result of the call).

This implementation iterates over the collection looking for the specified element. If it finds the element, it removes the element from the collection using the iterator's remove method.

Note that this implementation throws an `UnsupportedOperationException` if the iterator returned by this collection's iterator method does not implement the remove method and this collection contains the specified object.

Parameters

value - element to be removed from this collection, if present

Returns

true if an element was removed as a result of this call

Exceptions

UnsupportedOperationException if the remove operation is not supported by this collection.

IllegalArgumentException If the value is not a valid entry for this **Collection** (p. 982).

Reimplemented from **decaf::util::AbstractCollection**< **E** > (p. 127).

6.649.3.10 `template<typename E> virtual std::size_t decaf::util::StlSet< E >::size
() const [inline, virtual]`

Returns

The number of elements in this set.

Implements **decaf::util::Collection**< **E** > (p. 990).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/StlSet.h`

6.650 activemq::wireformat::stomp::StompCommandConstants Class Reference

```
#include <src/main/activemq/wireformat/stomp/StompCommandConstants.h>
```

Static Public Attributes

- static const std::string **CONNECT**
- static const std::string **CONNECTED**
- static const std::string **DISCONNECT**
- static const std::string **SUBSCRIBE**
- static const std::string **UNSUBSCRIBE**
- static const std::string **MESSAGE**
- static const std::string **SEND**
- static const std::string **BEGIN**
- static const std::string **COMMIT**
- static const std::string **ABORT**
- static const std::string **ACK**
- static const std::string **ERROR_CMD**
- static const std::string **RECEIPT**
- static const std::string **HEADER_DESTINATION**
- static const std::string **HEADER_TRANSACTIONID**

- static const std::string **HEADER_CONTENTLENGTH**
- static const std::string **HEADER_SESSIONID**
- static const std::string **HEADER_RECEIPT_REQUIRED**
- static const std::string **HEADER_RECEIPTID**
- static const std::string **HEADER_MESSAGEID**
- static const std::string **HEADER_ACK**
- static const std::string **HEADER_LOGIN**
- static const std::string **HEADER_PASSWORD**
- static const std::string **HEADER_CLIENT_ID**
- static const std::string **HEADER_MESSAGE**
- static const std::string **HEADER_CORRELATIONID**
- static const std::string **HEADER_REQUESTID**
- static const std::string **HEADER_RESPONSEID**
- static const std::string **HEADER_EXPIRES**
- static const std::string **HEADER_PERSISTENT**
- static const std::string **HEADER_REPLYTO**
- static const std::string **HEADER_TYPE**
- static const std::string **HEADER_DISPATCH_ASYNC**
- static const std::string **HEADER_EXCLUSIVE**
- static const std::string **HEADER_MAXPENDINGMSGLIMIT**
- static const std::string **HEADER_NOLOCAL**
- static const std::string **HEADER_PREFETCHSIZE**
- static const std::string **HEADER_JMSPRIORITY**
- static const std::string **HEADER_CONSUMERPRIORITY**
- static const std::string **HEADER_RETROACTIVE**
- static const std::string **HEADER_SUBSCRIPTIONNAME**
- static const std::string **HEADER_OLDSUBSCRIPTIONNAME**
- static const std::string **HEADER_TIMESTAMP**
- static const std::string **HEADER_REDELIVERED**
- static const std::string **HEADER_REDELIVERYCOUNT**
- static const std::string **HEADER_SELECTOR**
- static const std::string **HEADER_ID**
- static const std::string **HEADER_SUBSCRIPTION**
- static const std::string **HEADER_TRANSFORMATION**
- static const std::string **HEADER_TRANSFORMATION_ERROR**
- static const std::string **ACK_CLIENT**
- static const std::string **ACK_AUTO**
- static const std::string **ACK_INDIVIDUAL**
- static const std::string **TEXT**
- static const std::string **BYTES**
- static const std::string **QUEUE_PREFIX**
- static const std::string **TOPIC_PREFIX**
- static const std::string **TEMPQUEUE_PREFIX**
- static const std::string **TEMPTOPIC_PREFIX**

6.650.1 Field Documentation

- 6.650.1.1** `const std::string activemq::wireformat::stomp::StompCommandConstants::ABORT`
[static]
- 6.650.1.2** `const std::string activemq::wireformat::stomp::StompCommandConstants::ACK`
[static]
- 6.650.1.3** `const std::string activemq::wireformat::stomp::StompCommandConstants::ACK_AUTO`
[static]
- 6.650.1.4** `const std::string activemq::wireformat::stomp::StompCommandConstants::ACK_CLIENT` [static]
- 6.650.1.5** `const std::string activemq::wireformat::stomp::StompCommandConstants::ACK_INDIVIDUAL` [static]
- 6.650.1.6** `const std::string activemq::wireformat::stomp::StompCommandConstants::BEGIN`
[static]
- 6.650.1.7** `const std::string activemq::wireformat::stomp::StompCommandConstants::BYTES`
[static]
- 6.650.1.8** `const std::string activemq::wireformat::stomp::StompCommandConstants::COMMIT`
[static]
- 6.650.1.9** `const std::string activemq::wireformat::stomp::StompCommandConstants::CONNECT`
[static]
- 6.650.1.10** `const std::string activemq::wireformat::stomp::StompCommandConstants::CONNECTED`
[static]
- 6.650.1.11** `const std::string activemq::wireformat::stomp::StompCommandConstants::DISCONNECT`
[static]
- 6.650.1.12** `const std::string activemq::wireformat::stomp::StompCommandConstants::ERROR_CMD` [static]
- 6.650.1.13** `const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_ACK` [static]
- 6.650.1.14** `const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_CLIENT_ID` [static]
- 6.650.1.15** `const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_CONSUMERPRIORITY` [static]

- `src/main/activemq/wireformat/stomp/StompCommandConstants.h`

6.651 activemq::wireformat::stomp::StompFrame Class Reference

A Stomp-level message frame that encloses all messages to and from the broker.

```
#include <src/main/activemq/wireformat/stomp/StompFrame.h>
```

Public Member Functions

- **StompFrame** ()
Default constructor.
- virtual **~StompFrame** ()
Destruction.
- **StompFrame * clone** () const
Clone this message exactly, returns a new instance that the caller is required to delete.
- void **copy** (const **StompFrame** *src)
Copies the contents of the passed Frame to this one.
- void **setCommand** (const std::string &cmd)
Sets the command for this stomp frame.
- const std::string & **getCommand** () const
Accessor for this frame's command field.
- bool **hasProperty** (const std::string &name) const
Checks if the given property is present in the Frame.
- std::string **getProperty** (const std::string &name, const std::string &fallback="") const
Gets a property from this Frame's properties and returns it, or the default value given.
- std::string **removeProperty** (const std::string &name)
Gets and remove the property specified, if the property is not set, this method returns the empty string.
- void **setProperty** (const std::string &name, const std::string &value)
Sets the property given to the value specified in this Frame's Properties.
- **decaf::util::Properties** & **getProperties** ()
Gets access to the header properties for this frame.
- const **decaf::util::Properties** & **getProperties** () const
- const std::vector< unsigned char > & **getBody** () const
Accessor for the body data of this frame.

- `std::vector< unsigned char > & getBody ()`
Non-const version of the body accessor.
- `std::size_t getBodyLength () const`
Return the number of bytes contained in this frames body.
- `void setBody (const unsigned char *bytes, std::size_t numBytes)`
Sets the body data of this frame as a byte sequence.
- `void toStream (decaf::io::DataOutputStream *stream) const throw (decaf::io::IOException)`
Writes this Frame to an OuputStream in the Stomp Wire Format.
- `void fromStream (decaf::io::DataInputStream *stream) throw (decaf::io::IOException)`
Reads a Stop Frame from a DataInputStream in the Stomp Wire format.

6.651.1 Detailed Description

A Stomp-level message frame that encloses all messages to and from the broker.

6.651.2 Constructor & Destructor Documentation

6.651.2.1 `activemq::wireformat::stomp::StompFrame::StompFrame () [inline]`

Default constructor.

6.651.2.2 `virtual activemq::wireformat::stomp::StompFrame::~~StompFrame () [inline, virtual]`

Destruction.

6.651.3 Member Function Documentation

6.651.3.1 `StompFrame* activemq::wireformat::stomp::StompFrame::clone () const`

Clonse this message exactly, returns a new instance that the caller is required to delete.

Returns

new copy of this message

6.651.3.2 `void activemq::wireformat::stomp::StompFrame::copy (const StompFrame * src)`

Copies the contents of the passed Frame to this one.

Parameters

src - Frame to copy

6.651.3.3 void activemq::wireformat::stomp::StompFrame::fromStream (decaf::io::DataInputStream * *stream*) throw (decaf::io::IOException)

Reads a Stop Frame from a DataInputStream in the Stomp Wire format.

Parameters

stream - The stream to read the Frame from.

Exceptions

IOException if an error occurs while writing the Frame.

6.651.3.4 const std::vector<unsigned char>& activemq::wireformat::stomp::StompFrame::getBody () const [inline]

Accessor for the body data of this frame.

Returns

char pointer to body data

6.651.3.5 std::vector<unsigned char>& activemq::wireformat::stomp::StompFrame::getBody () [inline]

Non-const version of the body accessor.

6.651.3.6 std::size_t activemq::wireformat::stomp::StompFrame::getBodyLength () const [inline]

Return the number of bytes contained in this frames body.

Returns

Body bytes length.

6.651.3.7 const std::string& activemq::wireformat::stomp::StompFrame::getCommand () const [inline]

Accessor for this frame's command field.

6.651.3.8 `decaf::util::Properties& activemq::wireformat::stomp::StompFrame::getProperties () [inline]`

Gets access to the header properties for this frame.

Returns

the Properties object owned by this Frame

6.651.3.9 `const decaf::util::Properties& activemq::wireformat::stomp::StompFrame::getProperties () const [inline]`

6.651.3.10 `std::string activemq::wireformat::stomp::StompFrame::getProperty (const std::string & name, const std::string & fallback = "") const [inline]`

Gets a property from this Frame's properties and returns it, or the default value given.

Parameters

name - The name of the property to lookup

fallback - The default value to return if this value isn't set

Returns

string value of the property asked for.

6.651.3.11 `bool activemq::wireformat::stomp::StompFrame::hasProperty (const std::string & name) const [inline]`

Checks if the given property is present in the Frame.

Parameters

name - The name of the property to check for.

6.651.3.12 `std::string activemq::wireformat::stomp::StompFrame::removeProperty (const std::string & name) [inline]`

Gets and remove the property specified, if the property is not set, this method returns the empty string.

Parameters

name - the Name of the property to get and return.

6.651.3.13 `void activemq::wireformat::stomp::StompFrame::setBody (const unsigned char * bytes, std::size_t numBytes)`

Sets the body data of this frame as a byte sequence.

Parameters

bytes The byte buffer to be set in the body.

numBytes The number of bytes in the buffer.

6.651.3.14 `void activemq::wireformat::stomp::StompFrame::setCommand (const std::string & cmd) [inline]`

Sets the command for this stomp frame.

Parameters

cmd command The command to be set.

6.651.3.15 `void activemq::wireformat::stomp::StompFrame::setProperty (const std::string & name, const std::string & value) [inline]`

Sets the property given to the value specified in this Frame's Properties.

Parameters

name - Name of the property.

value - Value to set the property to.

6.651.3.16 `void activemq::wireformat::stomp::StompFrame::toStream (decaf::io::DataOutputStream * stream) const throw (decaf::io::IOException)`

Writes this Frame to an OutputStream in the Stomp Wire Format.

Parameters

stream - The stream to write the Frame to.

Exceptions

IOException if an error occurs while reading the Frame.

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/stomp/StompFrame.h`

6.652 activemq::wireformat::stomp::StompHelper Class Reference

Utility Methods used when marshaling to and from StompFrame's.

```
#include <src/main/activemq/wireformat/stomp/StompHelper.h>
```

Public Member Functions

- **StompHelper** ()
- virtual **~StompHelper** ()
- void **convertProperties** (const **Pointer**< **StompFrame** > &frame, const **Pointer**< **Message** > &message)

Converts the Headers in a Stomp Frame into Headers in the given Message Command.
- void **convertProperties** (const **Pointer**< **Message** > &message, const **Pointer**< **StompFrame** > &frame)

*Converts the Properties in a Message Command to Valid Headers and Properties in the **StompFrame** (p. 2874).*
- **Pointer**< **ActiveMQDestination** > **convertDestination** (const std::string &destination)

Converts from a Stomp Destination to an ActiveMQDestination.
- std::string **convertDestination** (const **Pointer**< **ActiveMQDestination** > &destination)

Converts from a ActiveMQDestination to a Stomp Destination Name.
- std::string **convertMessageId** (const **Pointer**< **MessageId** > &messageId)

Converts a MessageId instance to a Stomp MessageId String.
- **Pointer**< **MessageId** > **convertMessageId** (const std::string &messageId)

Converts a Stomp MessageId string to a MessageId.
- std::string **convertConsumerId** (const **Pointer**< **ConsumerId** > &consumerId)

Converts a ConsumerId instance to a Stomp ConsumerId String.
- **Pointer**< **ConsumerId** > **convertConsumerId** (const std::string &consumerId)

Converts a Stomp ConsumerId string to a ConsumerId.
- std::string **convertProducerId** (const **Pointer**< **ProducerId** > &producerId)

Converts a ProducerId instance to a Stomp ProducerId String.
- **Pointer**< **ProducerId** > **convertProducerId** (const std::string &producerId)

Converts a Stomp ProducerId string to a ProducerId.
- std::string **convertTransactionId** (const **Pointer**< **TransactionId** > &transactionId)

Converts a TransactionId instance to a Stomp TransactionId String.
- **Pointer**< **TransactionId** > **convertTransactionId** (const std::string &transactionId)

Converts a Stomp TransactionId string to a TransactionId.

6.652.1 Detailed Description

Utility Methods used when marshaling to and from StompFrame's.

Since

3.0

6.652.2 Constructor & Destructor Documentation

6.652.2.1 `activemq::wireformat::stomp::StompHelper::StompHelper ()` [inline]

6.652.2.2 `virtual activemq::wireformat::stomp::StompHelper::~~StompHelper ()`
[inline, virtual]

6.652.3 Member Function Documentation

6.652.3.1 `std::string activemq::wireformat::stomp::StompHelper::convertConsumerId (const Pointer< ConsumerId > & consumerId)`

Converts a ConsumerId instance to a Stomp ConsumerId String.

Parameters

consumerId - the Consumer instance to convert.

Returns

a Stomp Consumer Id String.

6.652.3.2 `Pointer<ConsumerId> activemq::wireformat::stomp::StompHelper::convertConsumerId (const std::string & consumerId)`

Converts a Stomp ConsumerId string to a ConsumerId.

Parameters

consumerId - the String Consumer Id to convert.

Returns

Pointer to a new ConsumerId.

6.652.3.3 `std::string activemq::wireformat::stomp::StompHelper::convertDestination (const Pointer< ActiveMQDestination > & destination)`

Converts from a ActiveMQDestination to a Stomp Destination Name.

Parameters

destination - The ActiveMQDestination to Convert

Returns

the Stomp String name that defines the destination.

6.652.3.4 `Pointer<ActiveMQDestination> activemq::wireformat::stomp::StompHelper::convertDestination (const std::string & destination)`

Converts from a Stomp Destination to an ActiveMQDestination.

Parameters

destination - The Stomp Destination name string.

Returns

Pointer to a new ActiveMQDestination.

6.652.3.5 `std::string activemq::wireformat::stomp::StompHelper::convertMessageId (const Pointer< MessageId > & messageId)`

Converts a MessageId instance to a Stomp MessageId String.

Parameters

messageId - the MessageId instance to convert.

Returns

a Stomp Message Id String.

6.652.3.6 `Pointer<MessageId> activemq::wireformat::stomp::StompHelper::convertMessageId (const std::string & messageId)`

Converts a Stomp MessageId string to a MessageId.

Parameters

messageId - the String message Id to convert.

Returns

Pointer to a new MessageId.

6.652.3.7 `std::string activemq::wireformat::stomp::StompHelper::convertProducerId (const Pointer< ProducerId > & producerId)`

Converts a ProducerId instance to a Stomp ProducerId String.

Parameters

producerId - the Producer instance to convert.

Returns

a Stomp Producer Id String.

6.652.3.8 `Pointer<ProducerId> activemq::wireformat::stomp::StompHelper::convertProducerId (const std::string & producerId)`

Converts a Stomp ProducerId string to a ProducerId.

Parameters

producerId - the String Producer Id to convert.

Returns

Pointer to a new ProducerId.

6.652.3.9 `void activemq::wireformat::stomp::StompHelper::convertProperties (const Pointer< Message > & message, const Pointer< StompFrame > & frame)`

Converts the Properties in a Message Command to Valid Headers and Properties in the **StompFrame** (p.2874).

Parameters

message - The message to move the Headers to.

frame - The frame to extract headers from.

6.652.3.10 `void activemq::wireformat::stomp::StompHelper::convertProperties (const Pointer< StompFrame > & frame, const Pointer< Message > & message)`

Converts the Headers in a Stomp Frame into Headers in the given Message Command.

Parameters

frame - The frame to extract headers from.

message - The message to move the Headers to.

6.652.3.11 `Pointer<TransactionId> activemq::wireformat::stomp::StompHelper::convertTransactionId (const std::string & transactionId)`

Converts a Stomp TransactionId string to a TransactionId.

Parameters

transactionId - the String Transaction Id to convert.

Returns

Pointer to a new TransactionId.

6.652.3.12 `std::string activemq::wireformat::stomp::StompHelper::convertTransactionId (const Pointer< TransactionId > & transactionId)`

Converts a TransactionId instance to a Stomp TransactionId String.

Parameters

transactionId - the Transaction instance to convert.

Returns

a Stomp Transaction Id String.

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/stomp/StompHelper.h`

6.653 `activemq::wireformat::stomp::StompWireFormat` Class Reference

```
#include <src/main/activemq/wireformat/stomp/StompWireFormat.h>
```

Inheritance diagram for `activemq::wireformat::stomp::StompWireFormat`:

Public Member Functions

- `StompWireFormat ()`
- `virtual ~StompWireFormat ()`
- `virtual void marshal (const Pointer< commands::Command > &command, const activemq::transport::Transport *transport, decaf::io::DataOutputStream *out) throw (decaf::io::IOException)`
Stream based marshaling of a Command, this method blocks until the entire Command has been written out to the output stream.
- `virtual Pointer< commands::Command > unmarshal (const activemq::transport::Transport *transport, decaf::io::DataInputStream *in) throw (decaf::io::IOException)`
Stream based un-marshaling, blocks on reads on the input stream until a complete command has been read and unmarshaled into the correct form.
- `virtual void setVersion (int version AMQCPP_UNUSED)`
Set the Version.
- `virtual int getVersion () const`
Get the Version.
- `virtual bool inReceive () const`
Is there a Message being unmarshaled?

- virtual bool **hasNegotiator** () const

Returns true if this **WireFormat** (p. 3148) has a Negotiator that needs to wrap the Transport that uses it.

- virtual **Pointer< transport::Transport > createNegotiator** (const **Pointer< transport::Transport >** &transport) throw (decaf::lang::exceptions::UnsupportedOperationException)

If the Transport Provides a Negotiator this method will create and return a news instance of the Negotiator.

6.653.1 Constructor & Destructor Documentation

6.653.1.1 **activemq::wireformat::stomp::StompWireFormat::StompWireFormat** ()

6.653.1.2 **virtual**
activemq::wireformat::stomp::StompWireFormat::~~StompWireFormat () [virtual]

6.653.2 Member Function Documentation

6.653.2.1 **virtual Pointer<transport::Transport> activemq::wireformat::stomp::StompWireFormat::createNegotiator** (const **Pointer< transport::Transport >** & *transport*) throw (decaf::lang::exceptions::UnsupportedOperationException) [virtual]

If the Transport Provides a Negotiator this method will create and return a news instance of the Negotiator.

Returns

new instance of a **WireFormatNegotiator** (p. 3182).

Exceptions

UnsupportedOperationException if the **WireFormat** (p. 3148) doesn't have a Negotiator.

Implements **activemq::wireformat::WireFormat** (p. 3149).

6.653.2.2 **virtual int activemq::wireformat::stomp::StompWireFormat::getVersion** () const [inline, virtual]

Get the Version.

Returns

the version of the wire format

Implements **activemq::wireformat::WireFormat** (p. 3149).

6.653.2.3 `virtual bool activemq::wireformat::stomp::StompWireFormat::hasNegotiator () const`
`[inline, virtual]`

Returns true if this **WireFormat** (p. 3148) has a Negotiator that needs to wrap the Transport that uses it.

Returns

true if the **WireFormat** (p. 3148) provides a Negotiator.

Implements **activemq::wireformat::WireFormat** (p. 3149).

6.653.2.4 `virtual bool activemq::wireformat::stomp::StompWireFormat::inReceive () const`
`[inline, virtual]`

Is there a Message being unmarshaled?

Returns

true while in the doUnmarshal method.

Implements **activemq::wireformat::WireFormat** (p. 3150).

6.653.2.5 `virtual void activemq::wireformat::stomp::StompWireFormat::marshal (const Pointer< commands::Command > & command, const activemq::transport::Transport * transport, decaf::io::DataOutputStream * out) throw (decaf::io::IOException)`
`[virtual]`

Stream based marshaling of a Command, this method blocks until the entire Command has been written out to the output stream.

Parameters

command The Command to Marshal to the output stream.

transport The Transport that initiated this marshal call.

out The output stream to write the command to.

Exceptions

IOException

Implements **activemq::wireformat::WireFormat** (p. 3150).

6.653.2.6 `virtual void activemq::wireformat::stomp::StompWireFormat::setVersion (int version AMQCPP_UNUSED)`
`[inline, virtual]`

Set the Version.

Parameters

the version of the wire format

Implements **activemq::wireformat::WireFormat** (p. 3150).

6.653.2.7 virtual Pointer<commands::Command> activemq::wireformat::stomp::StompWireFormat::unmarshal (const activemq::transport::Transport * *transport*, decaf::io::DataInputStream * *in*) throw (decaf::io::IOException)
[virtual]

Stream based un-marshaling, blocks on reads on the input stream until a complete command has been read and unmarshaled into the correct form.

Returns a Pointer to the newly unmarshaled Command.

Parameters

transport - Pointer to the transport that is making this request.

in - the input stream to read the command from.

Returns

the newly marshaled Command, caller owns the pointer

Exceptions

IOException

Implements **activemq::wireformat::WireFormat** (p. 3151).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/stomp/StompWireFormat.h

6.654 activemq::wireformat::stomp::StompWireFormatFactory Class Reference

Factory used to create the Stomp Wire Format instance.

```
#include <src/main/activemq/wireformat/stomp/StompWireFormatFactory.h>
```

Inheritance diagram for activemq::wireformat::stomp::StompWireFormatFactory:

Public Member Functions

- **StompWireFormatFactory** ()
- virtual ~**StompWireFormatFactory** ()
- virtual **Pointer**< **WireFormat** > **createWireFormat** (const decaf::util::Properties &properties) throw (decaf::lang::exceptions::IllegalStateException)

*Creates a new **WireFormat** (p. 3148) Object passing it a set of properties from which it can obtain any optional settings.*

6.654.1 Detailed Description

Factory used to create the Stomp Wire Format instance.

6.654.2 Constructor & Destructor Documentation

- 6.654.2.1** `activemq::wireformat::stomp::StompWireFormatFactory::StompWireFormatFactory ()` [inline]
- 6.654.2.2** `virtual activemq::wireformat::stomp::StompWireFormatFactory::~~StompWireFormatFactory ()` [inline, virtual]

6.654.3 Member Function Documentation

- 6.654.3.1** `virtual Pointer<WireFormat> activemq::wireformat::stomp::StompWireFormatFactory::createWireFormat (const decaf::util::Properties & properties) throw (decaf::lang::exceptions::IllegalStateException)` [virtual]

Creates a new **WireFormat** (p. 3148) Object passing it a set of properties from which it can obtain any optional settings.

Parameters

properties - the Properties for this **WireFormat** (p. 3148)

Implements **activemq::wireformat::WireFormatFactory** (p. 3152).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/stomp/StompWireFormatFactory.h`

6.655 cms::Stoppable Class Reference

Interface for a class that implements the stop method.

```
#include <src/main/cms/Stoppable.h>
```

Inheritance diagram for cms::Stoppable:

Public Member Functions

- `virtual ~Stoppable ()`
- `virtual void stop ()=0 throw (CMSException)`
Stops this service.

6.655.1 Detailed Description

Interface for a class that implements the stop method. An object that implements this interface implies that it will halt all operations that result in events being propagated to external users, internally the Object can continue to process data but not events will be generated to clients and methods that return data will not return valid results until the object is started again.

Since

1.0

6.655.2 Constructor & Destructor Documentation

6.655.2.1 virtual cms::Stoppable::~~Stoppable () [inline, virtual]

6.655.3 Member Function Documentation

6.655.3.1 virtual void cms::Stoppable::stop () throw (CMSException) [pure virtual]

Stops this service.

Exceptions

CMSException (p. 960) - if an internal error occurs while stopping the Service.

The documentation for this class was generated from the following file:

- src/main/cms/Stoppable.h

6.656 decaf::util::logging::StreamHandler Class Reference

```
#include <src/main/decaf/util/logging/StreamHandler.h>
```

Inheritance diagram for decaf::util::logging::StreamHandler:

Public Member Functions

- **StreamHandler** ()
*Create a **StreamHandler** (p. 2888), with no current output stream.*
- **StreamHandler** (io::OutputStream *stream, Formatter *formatter)
*Create a **StreamHandler** (p. 2888), with no current output stream.*
- virtual ~**StreamHandler** ()
- virtual void **close** () throw (decaf::io::IOException)
Close the current output stream.
- virtual void **flush** ()
Flush the Handler's output, clears any buffers.
- virtual void **publish** (const **LogRecord** &record)
*Publish the Log Record to this **Handler** (p. 1604).*
- virtual void **isLoggable** (const **LogRecord** &record)

Check if this **Handler** (p. 1604) would actually log a given **LogRecord** (p. 1928).

- virtual void **setFilter** (const **Filter** *filter)
Sets the **Filter** (p. 1527) that this **Handler** (p. 1604) uses to filter Log Records.
- virtual const **Filter** * **getFilter** ()
Gets the **Filter** (p. 1527) that this **Handler** (p. 1604) uses to filter Log Records.
- virtual void **setLevel** (**Level** level)
Set (p. 2729) the log level specifying which message levels will be logged by this **Handler** (p. 1604).
- virtual **Level** **getLevel** ()
Get the log level specifying which message levels will be logged by this **Handler** (p. 1604).
- virtual void **setFormatter** (const **Formatter** *formatter)
Sets the **Formatter** (p. 1595) used by this **Handler** (p. 1604).
- virtual const **Formatter** * **getFormatter** ()
Gets the **Formatter** (p. 1595) used by this **Handler** (p. 1604).
- virtual **io::OutputStream** * **getOutputStream** () const
Gets the output Stream that this **Handler** (p. 1604) is using.

6.656.1 Constructor & Destructor Documentation

6.656.1.1 decaf::util::logging::StreamHandler::StreamHandler () [inline]

Create a **StreamHandler** (p. 2888), with no current output stream.

6.656.1.2 decaf::util::logging::StreamHandler::StreamHandler (io::OutputStream * stream, Formatter * formatter) [inline]

Create a **StreamHandler** (p. 2888), with no current output stream.

6.656.1.3 virtual decaf::util::logging::StreamHandler::~~StreamHandler () [inline, virtual]

References DECAF_CATCH_NOTHROW, and DECAF_CATCHALL_NOTHROW.

6.656.2 Member Function Documentation

6.656.2.1 virtual void decaf::util::logging::StreamHandler::close () throw (decaf::io::IOException) [inline, virtual]

Close the current output stream.

The close method will perform a flush and then close the **Handler** (p. 1604). After close has been called this **Handler** (p. 1604) should no longer be used. Method calls may either be silently ignored or may throw runtime exceptions.

Exceptions

IOException

Implements **decaf::io::Closeable** (p. 951).

6.656.2.2 virtual void decaf::util::logging::StreamHandler::flush () [inline, virtual]

Flush the Handler's output, clears any buffers.

Implements **decaf::util::logging::Handler** (p. 1605).

6.656.2.3 virtual const Filter* decaf::util::logging::StreamHandler::getFilter () [inline, virtual]

Gets the **Filter** (p. 1527) that this **Handler** (p. 1604) uses to filter Log Records.

Returns

Filter (p. 1527) derived instance

Implements **decaf::util::logging::Handler** (p. 1606).

6.656.2.4 virtual const Formatter* decaf::util::logging::StreamHandler::getFormatter () [inline, virtual]

Gets the **Formatter** (p. 1595) used by this **Handler** (p. 1604).

Returns

currently configured **Formatter** (p. 1595) derived instance

Implements **decaf::util::logging::Handler** (p. 1606).

6.656.2.5 virtual Level decaf::util::logging::StreamHandler::getLevel () [inline, virtual]

Get the log level specifying which message levels will be logged by this **Handler** (p. 1604).

Returns

Currently set Level enumeration value

Implements **decaf::util::logging::Handler** (p. 1606).

6.656.2.6 virtual io::OutputStream* decaf::util::logging::StreamHandler::getOutputStream () const [inline, virtual]

Gets the output Stream that this **Handler** (p. 1604) is using.

Returns

OutputStream pointer used by this handler.

6.656.2.7 `virtual void decaf::util::logging::StreamHandler::isLoggable (const LogRecord & record) [inline, virtual]`

Check if this **Handler** (p. 1604) would actually log a given **LogRecord** (p. 1928).

Parameters

record LogRecord (p. 1928) to check

Implements **decaf::util::logging::Handler** (p. 1606).

6.656.2.8 `virtual void decaf::util::logging::StreamHandler::publish (const LogRecord & record) [inline, virtual]`

Publish the Log Record to this **Handler** (p. 1604).

Parameters

record The LogRecord (p. 1928) to Publish

Implements **decaf::util::logging::Handler** (p. 1606).

References DECAF_CATCH_RETHROW, and DECAF_CATCHALL_THROW.

6.656.2.9 `virtual void decaf::util::logging::StreamHandler::setFilter (const Filter * filter) [inline, virtual]`

Sets the **Filter** (p. 1527) that this **Handler** (p. 1604) uses to filter Log Records.

Parameters

filter Filter (p. 1527) derived instance

Implements **decaf::util::logging::Handler** (p. 1607).

6.656.2.10 `virtual void decaf::util::logging::StreamHandler::setFormatter (const Formatter * formatter) [inline, virtual]`

Sets the Formatter (p. 1595) used by this **Handler** (p. 1604).

Parameters

formatter Formatter (p. 1595) derived instance

Implements **decaf::util::logging::Handler** (p. 1607).

6.656.2.11 `virtual void decaf::util::logging::StreamHandler::setLevel (Level level) [inline, virtual]`

Set (p. 2729) the log level specifying which message levels will be logged by this **Handler** (p. 1604).

The intention is to allow developers to turn on verbose logging, but to limit the messages that are sent to certain Handlers.

Parameters

level Level enumeration value

Implements **decaf::util::logging::Handler** (p. 1607).

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/**StreamHandler.h**

6.657 cms::StreamMessage Class Reference

Interface for a **StreamMessage** (p. 2892).

`#include <src/main/cms/StreamMessage.h>`

Inheritance diagram for cms::StreamMessage:

Public Member Functions

- virtual `~StreamMessage ()`
- virtual `bool readBoolean () const =0 throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException)`
Reads a Boolean from the Stream message stream.
- virtual `void writeBoolean (bool value)=0 throw (cms::MessageNotWriteableException, cms::CMSException)`
Writes a boolean to the Stream message stream as a 1-byte value.
- virtual `unsigned char readByte () const =0 throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException)`
Reads a Byte from the Stream message stream.
- virtual `void writeByte (unsigned char value)=0 throw (cms::MessageNotWriteableException, cms::CMSException)`
Writes a byte to the Stream message stream as a 1-byte value.
- virtual `std::size_t readBytes (std::vector< unsigned char > &value) const =0 throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException)`
Reads a byte array from the Stream message stream.

- virtual void **writeBytes** (const std::vector< unsigned char > &value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)

Writes a byte array to the Stream message stream using the vector size as the number of bytes to write.

- virtual std::size_t **readBytes** (unsigned char *buffer, std::size_t length) const =0 throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException)

Reads a portion of the Stream message stream.

- virtual void **writeBytes** (const unsigned char *value, std::size_t offset, std::size_t length)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)

Writes a portion of a byte array to the Stream message stream.

- virtual char **readChar** () const =0 throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException)

Reads a Char from the Stream message stream.

- virtual void **writeChar** (char value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)

Writes a char to the Stream message stream as a 1-byte value.

- virtual float **readFloat** () const =0 throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException)

Reads a 32 bit float from the Stream message stream.

- virtual void **writeFloat** (float value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)

Writes a float to the Stream message stream as a 4 byte value.

- virtual double **readDouble** () const =0 throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException)

Reads a 64 bit double from the Stream message stream.

- virtual void **writeDouble** (double value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)

Writes a double to the Stream message stream as a 8 byte value.

- virtual short **readShort** () const =0 throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException)

Reads a 16 bit signed short from the Stream message stream.

- virtual void **writeShort** (short value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)

Writes a signed short to the Stream message stream as a 2 byte value.

- virtual unsigned short **readUnsignedShort** () const =0 throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException)
Reads a 16 bit unsigned short from the Stream message stream.
- virtual void **writeUnsignedShort** (unsigned short value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)
Writes a unsigned short to the Stream message stream as a 2 byte value.
- virtual int **readInt** () const =0 throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException)
Reads a 32 bit signed integer from the Stream message stream.
- virtual void **writeInt** (int value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)
Writes a signed int to the Stream message stream as a 4 byte value.
- virtual long long **readLong** () const =0 throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException)
Reads a 64 bit long from the Stream message stream.
- virtual void **writeLong** (long long value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)
Writes a long long to the Stream message stream as a 8 byte value.
- virtual std::string **readString** () const =0 throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSEException)
Reads an ASCII String from the Stream message stream.
- virtual void **writeString** (const std::string &value)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)
Writes an ASCII String to the Stream message stream.

6.657.1 Detailed Description

Interface for a **StreamMessage** (p. 2892). The stream Messages provides a **Message** (p. 2036) type whose body is a stream of self describing primitive types. The primitive values are read and written using accessors specific to the given types.

StreamMessage (p. 2892) objects support the following conversion table. The marked cases must be supported. The unmarked cases must throw a **CMSEException** (p. 960). The string-to-primitive conversions may throw a runtime exception if the primitive's `valueOf()` method does not accept it as a valid String representation of the primitive.

A value written as the row type can be read as the column type.

		boolean	byte	short	char	int	long	float	double	String	byte[]

boolean	X								X
byte		X	X		X	X			X
short			X		X	X			X
char				X					X
int					X	X			X
long						X			X
float							X	X	X
double								X	X
String	X	X	X		X	X	X	X	X
byte[]									X

Since

1.3

6.657.2 Constructor & Destructor Documentation

6.657.2.1 `virtual cms::StreamMessage::~StreamMessage () [inline, virtual]`

6.657.3 Member Function Documentation

6.657.3.1 `virtual bool cms::StreamMessage::readBoolean () const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException) [pure virtual]`

Reads a Boolean from the Stream message stream.

Returns

boolean value from stream

Exceptions

CMSException (p. 960) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. 2140) - if unexpected end of message stream has been reached.

MessageFormatException (p. 2141) - if this type conversion is invalid.

MessageNotReadableException (p. 2187) - if the message is in write-only mode.

6.657.3.2 `virtual unsigned char cms::StreamMessage::readByte () const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException) [pure virtual]`

Reads a Byte from the Stream message stream.

Returns

unsigned char value from stream

Exceptions

CMSException (p. 960) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. 2140) - if unexpected end of message stream has been reached.

MessageFormatException (p. 2141) - if this type conversion is invalid.

MessageNotReadableException (p. 2187) - if the message is in write-only mode.

6.657.3.3 `virtual std::size_t cms::StreamMessage::readBytes (std::vector< unsigned char > & value) const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException) [pure virtual]`

Reads a byte array from the Stream message stream.

If the length of vector value is less than the number of bytes remaining to be read from the stream, the vector should be filled. A subsequent call reads the next increment, and so on.

If the number of bytes remaining in the stream is less than the length of vector value, the bytes should be read into the vector. The return value of the total number of bytes read will be less than the length of the vector, indicating that there are no more bytes left to be read from the stream. The next read of the stream returns -1.

Parameters

value buffer to place data in

Returns

the total number of bytes read into the buffer, or -1 if there is no more data because the end of the stream has been reached

Exceptions

CMSException (p. 960) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. 2140) - if unexpected end of message stream has been reached.

MessageFormatException (p. 2141) - if this type conversion is invalid.

MessageNotReadableException (p. 2187) - if the message is in write-only mode.

6.657.3.4 `virtual std::size_t cms::StreamMessage::readBytes (unsigned char * buffer, std::size_t length) const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException) [pure virtual]`

Reads a portion of the Stream message stream.

If the length of array value is less than the number of bytes remaining to be read from the stream, the array should be filled. A subsequent call reads the next increment, and so on.

If the number of bytes remaining in the stream is less than the length of array value, the bytes should be read into the array. The return value of the total number of bytes read will be less than

the length of the array, indicating that there are no more bytes left to be read from the stream. The next read of the stream returns -1.

If length is negative, or length is greater than the length of the array value, then an **CMSException** (p. 960) is thrown. No bytes will be read from the stream for this exception case.

Parameters

buffer the buffer into which the data is read

length the number of bytes to read; must be less than or equal to value.length

Returns

the total number of bytes read into the buffer, or -1 if there is no more data because the end of the stream has been reached

Exceptions

CMSException (p. 960) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. 2140) - if unexpected end of message stream has been reached.

MessageFormatException (p. 2141) - if this type conversion is invalid.

MessageNotReadableException (p. 2187) - if the message is in write-only mode.

6.657.3.5 `virtual char cms::StreamMessage::readChar () const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException) [pure virtual]`

Reads a Char from the Stream message stream.

Returns

char value from stream

Exceptions

CMSException (p. 960) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. 2140) - if unexpected end of message stream has been reached.

MessageFormatException (p. 2141) - if this type conversion is invalid.

MessageNotReadableException (p. 2187) - if the message is in write-only mode.

6.657.3.6 `virtual double cms::StreamMessage::readDouble () const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException) [pure virtual]`

Reads a 64 bit double from the Stream message stream.

Returns

double value from stream

Exceptions

CMSException (p. 960) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. 2140) - if unexpected end of message stream has been reached.

MessageFormatException (p. 2141) - if this type conversion is invalid.

MessageNotReadableException (p. 2187) - if the message is in write-only mode.

6.657.3.7 virtual float cms::StreamMessage::readFloat () const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException) [pure virtual]

Reads a 32 bit float from the Stream message stream.

Returns

double value from stream

Exceptions

CMSException (p. 960) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. 2140) - if unexpected end of message stream has been reached.

MessageFormatException (p. 2141) - if this type conversion is invalid.

MessageNotReadableException (p. 2187) - if the message is in write-only mode.

6.657.3.8 virtual int cms::StreamMessage::readInt () const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException) [pure virtual]

Reads a 32 bit signed integer from the Stream message stream.

Returns

int value from stream

Exceptions

CMSException (p. 960) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. 2140) - if unexpected end of message stream has been reached.

MessageFormatException (p. 2141) - if this type conversion is invalid.

MessageNotReadableException (p. 2187) - if the message is in write-only mode.

6.657.3.9 `virtual long long cms::StreamMessage::readLong () const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException) [pure virtual]`

Reads a 64 bit long from the Stream message stream.

Returns

long long value from stream

Exceptions

CMSException (p. 960) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. 2140) - if unexpected end of message stream has been reached.

MessageFormatException (p. 2141) - if this type conversion is invalid.

MessageNotReadableException (p. 2187) - if the message is in write-only mode.

6.657.3.10 `virtual short cms::StreamMessage::readShort () const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException) [pure virtual]`

Reads a 16 bit signed short from the Stream message stream.

Returns

short value from stream

Exceptions

CMSException (p. 960) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. 2140) - if unexpected end of message stream has been reached.

MessageFormatException (p. 2141) - if this type conversion is invalid.

MessageNotReadableException (p. 2187) - if the message is in write-only mode.

6.657.3.11 `virtual std::string cms::StreamMessage::readString () const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException) [pure virtual]`

Reads an ASCII String from the Stream message stream.

Returns

String from stream

Exceptions

CMSException (p. 960) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. 2140) - if unexpected end of message stream has been reached.

MessageFormatException (p. 2141) - if this type conversion is invalid.

MessageNotReadableException (p. 2187) - if the message is in write-only mode.

6.657.3.12 virtual unsigned short cms::StreamMessage::readUnsignedShort () const throw (cms::MessageEOFException, cms::MessageFormatException, cms::MessageNotReadableException, cms::CMSException) [pure virtual]

Reads a 16 bit unsigned short from the Stream message stream.

Returns

unsigned short value from stream

Exceptions

CMSException (p. 960) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. 2140) - if unexpected end of message stream has been reached.

MessageFormatException (p. 2141) - if this type conversion is invalid.

MessageNotReadableException (p. 2187) - if the message is in write-only mode.

6.657.3.13 virtual void cms::StreamMessage::writeBoolean (bool value) throw (cms::MessageNotWriteableException, cms::CMSException) [pure virtual]

Writes a boolean to the Stream message stream as a 1-byte value.

The value true is written as the value (byte)1; the value false is written as the value (byte)0.

Parameters

value boolean to write to the stream

Exceptions

CMSException (p. 960) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. 2188) - if the message is in read-only mode.

6.657.3.14 `virtual void cms::StreamMessage::writeByte (unsigned char value)
throw (cms::MessageNotWriteableException, cms::CMSException)
[pure virtual]`

Writes a byte to the Stream message stream as a 1-byte value.

Parameters

value byte to write to the stream

Exceptions

CMSException (p. 960) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. 2188) - if the message is in read-only mode.

6.657.3.15 `virtual void cms::StreamMessage::writeBytes (const unsigned
char * value, std::size_t offset, std::size_t length) throw (cms::MessageNotWriteableException, cms::CMSException) [pure
virtual]`

Writes a portion of a byte array to the Stream message stream.

size as the number of bytes to write.

Parameters

value bytes to write to the stream

offset the initial offset within the byte array

length the number of bytes to use

Exceptions

CMSException (p. 960) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. 2188) - if the message is in read-only mode.

6.657.3.16 `virtual void cms::StreamMessage::writeBytes (const
std::vector< unsigned char > & value) throw (cms::MessageNotWriteableException, cms::CMSException) [pure
virtual]`

Writes a byte array to the Stream message stream using the vector size as the number of bytes to write.

Parameters

value bytes to write to the stream

Exceptions

CMSException (p. 960) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. 2188) - if the message is in read-only mode.

6.657.3.17 `virtual void cms::StreamMessage::writeChar (char value) throw (cms::MessageNotWriteableException, cms::CMSException) [pure virtual]`

Writes a char to the Stream message stream as a 1-byte value.

Parameters

value char to write to the stream

Exceptions

CMSException (p. 960) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. 2188) - if the message is in read-only mode.

6.657.3.18 `virtual void cms::StreamMessage::writeDouble (double value) throw (cms::MessageNotWriteableException, cms::CMSException) [pure virtual]`

Writes a double to the Stream message stream as a 8 byte value.

Parameters

value double to write to the stream

Exceptions

CMSException (p. 960) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. 2188) - if the message is in read-only mode.

6.657.3.19 `virtual void cms::StreamMessage::writeFloat (float value) throw (cms::MessageNotWriteableException, cms::CMSException) [pure virtual]`

Writes a float to the Stream message stream as a 4 byte value.

Parameters

value float to write to the stream

Exceptions

CMSException (p. 960) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. 2188) - if the message is in read-only mode.

6.657.3.20 `virtual void cms::StreamMessage::writeInt (int value) throw (cms::MessageNotWriteableException, cms::CMSException) [pure virtual]`

Writes a signed int to the Stream message stream as a 4 byte value.

Parameters

value signed int to write to the stream

Exceptions

CMSException (p. 960) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. 2188) - if the message is in read-only mode.

6.657.3.21 `virtual void cms::StreamMessage::writeLong (long long value) throw (cms::MessageNotWriteableException, cms::CMSException) [pure virtual]`

Writes a long long to the Stream message stream as a 8 byte value.

Parameters

value signed long long to write to the stream

Exceptions

CMSException (p. 960) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. 2188) - if the message is in read-only mode.

6.657.3.22 `virtual void cms::StreamMessage::writeShort (short value) throw (cms::MessageNotWriteableException, cms::CMSException) [pure virtual]`

Writes a signed short to the Stream message stream as a 2 byte value.

Parameters

value signed short to write to the stream

Exceptions

CMSException (p. 960) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. 2188) - if the message is in read-only mode.

6.657.3.23 virtual void cms::StreamMessage::writeString (const std::string & *value*) throw (cms::MessageNotWriteableException, cms::CMSException) [pure virtual]

Writes an ASCII String to the Stream message stream.

Parameters

value String to write to the stream

Exceptions

CMSException (p. 960) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. 2188) - if the message is in read-only mode.

6.657.3.24 virtual void cms::StreamMessage::writeUnsignedShort (unsigned short *value*) throw (cms::MessageNotWriteableException, cms::CMSException) [pure virtual]

Writes a unsigned short to the Stream message stream as a 2 byte value.

Parameters

value unsigned short to write to the stream

Exceptions

CMSException (p. 960) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. 2188) - if the message is in read-only mode.

The documentation for this class was generated from the following file:

- src/main/cms/StreamMessage.h

6.658 decaf::util::StringTokenizer Class Reference

```
#include <src/main/decaf/util/StringTokenizer.h>
```

Public Member Functions

- **StringTokenizer** (const std::string &str, const std::string &delim=" \t\n\r\f", bool returnDelims=false)

Constructs a string tokenizer for the specified string.

- virtual ~**StringTokenizer** ()
- virtual int **countTokens** () const

Calculates the number of times that this tokenizer's nextToken method can be called before it generates an exception.

- virtual bool **hasMoreTokens** () const
Tests if there are more tokens available from this tokenizer's string.
- virtual std::string **nextToken** () throw (lang::exceptions::NoSuchElementException)
Returns the next token from this string tokenizer.
- virtual std::string **nextToken** (const std::string &delim) throw (lang::exceptions::NoSuchElementException)
Returns the next token in this string tokenizer's string.
- virtual unsigned int **toArray** (std::vector< std::string > &array)
Grab all remaining tokens in the String and return them in the vector that is passed in by reference.
- virtual void **reset** (const std::string &str="", const std::string &delim="", bool returnDelims=false)
Resets the Tokenizer's position in the String to the Beginning calls to countToken and nextToken now start back at the beginning.

6.658.1 Constructor & Destructor Documentation

6.658.1.1 decaf::util::StringTokenizer::StringTokenizer (const std::string & str, const std::string & delim = " \t\n\r\f", bool returnDelims = false)

Constructs a string tokenizer for the specified string.

All characters in the delim argument are the delimiters for separating tokens.

If the returnDelims flag is true, then the delimiter characters are also returned as tokens. Each delimiter is returned as a string of length one. If the flag is false, the delimiter characters are skipped and only serve as separators between tokens.

Note that if delim is "", this constructor does not throw an exception. However, trying to invoke other methods on the resulting **StringTokenizer** (p. 2904) may result in an Exception.

Parameters

- str* - The string to tokenize
- delim* - String containing the delimiters
- returnDelims* - boolean indicating if the delimiters are returned as tokens

6.658.1.2 virtual decaf::util::StringTokenizer::~~StringTokenizer () [virtual]

6.658.2 Member Function Documentation

6.658.2.1 virtual int decaf::util::StringTokenizer::countTokens () const [virtual]

Calculates the number of times that this tokenizer's nextToken method can be called before it generates an exception.

The current position is not advanced.

Returns

Count of remaining tokens

6.658.2.2 `virtual bool decaf::util::StringTokenizer::hasMoreTokens () const`
[virtual]

Tests if there are more tokens available from this tokenizer's string.

Returns

true if there are more tokens remaining

6.658.2.3 `virtual std::string decaf::util::StringTokenizer::nextToken (const`
`std::string & delim) throw (lang::exceptions::NoSuchElementException`
`)` [virtual]

Returns the next token in this string tokenizer's string.

First, the set of characters considered to be delimiters by this **StringTokenizer** (p. 2904) object is changed to be the characters in the string *delim*. Then the next token in the string after the current position is returned. The current position is advanced beyond the recognized token. The new delimiter set remains the default after this call.

Parameters

delim - string containing the new set of delimiters

Returns

next string in the token list

Exceptions

NoSuchElementException

6.658.2.4 `virtual std::string decaf::util::StringTokenizer::nextToken () throw (`
`lang::exceptions::NoSuchElementException)` [virtual]

Returns the next token from this string tokenizer.

Returns

string value of next token

Exceptions

NoSuchElementException

6.658.2.5 `virtual void decaf::util::StringTokenizer::reset (const std::string & str = "", const std::string & delim = "", bool returnDelims = false) [virtual]`

Resets the Tokenizer's position in the String to the Beginning calls to countToken and nextToken now start back at the beginning.

This allows this object to be reused, the caller need not create a new instance every time a String needs tokenizing. If set the string param will reset the string that this Tokenizer is working on. If set to "" no change is made. If set the delim param will reset the string that this Tokenizer is using to tokenize the string. If set to "", no change is made. If set the return Delims will set if this Tokenizer will return delimiters as tokens. Defaults to false.

Parameters

str - New String to tokenize or "", defaults to ""

delim - New Delimiter String to use or "", defaults to ""

returnDelims - Should the Tokenizer return delimiters as Tokens, default false

6.658.2.6 `virtual unsigned int decaf::util::StringTokenizer::toArray (std::vector< std::string > & array) [virtual]`

Grab all remaining tokens in the String and return them in the vector that is passed in by reference.

Parameters

array - vector to place token strings in

Returns

number of string placed into the vector

The documentation for this class was generated from the following file:

- `src/main/decaf/util/StringTokenizer.h`

6.659 activemq::commands::SubscriptionInfo Class Reference

```
#include <src/main/activemq/commands/SubscriptionInfo.h>
```

Inheritance diagram for `activemq::commands::SubscriptionInfo`:

Public Member Functions

- `SubscriptionInfo ()`
- `virtual ~SubscriptionInfo ()`
- `virtual unsigned char getDataStructureType () const`

Get the unique identifier that this object and its own Marshaler share.

- virtual **SubscriptionInfo** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1372) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent.*
- virtual const std::string & **getClientId** () const
- virtual std::string & **getClientId** ()
- virtual void **setClientId** (const std::string &clientId)
- virtual const **Pointer**< **ActiveMQDestination** > & **getDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &destination)
- virtual const std::string & **getSelector** () const
- virtual std::string & **getSelector** ()
- virtual void **setSelector** (const std::string &selector)
- virtual const std::string & **getSubscriptionName** () const
- virtual std::string & **getSubscriptionName** ()
- virtual void **setSubscriptionName** (const std::string &subscriptionName)
- virtual const **Pointer**< **ActiveMQDestination** > & **getSubscribedDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getSubscribedDestination** ()
- virtual void **setSubscribedDestination** (const **Pointer**< **ActiveMQDestination** > &subscribedDestination)

Static Public Attributes

- static const unsigned char **ID_SUBSCRIPTIONINFO** = 55

Protected Member Functions

- **SubscriptionInfo** (const **SubscriptionInfo** &)
- **SubscriptionInfo** & **operator=** (const **SubscriptionInfo** &)

Protected Attributes

- std::string **clientId**
- **Pointer**< **ActiveMQDestination** > **destination**
- std::string **selector**
- std::string **subscriptionName**
- **Pointer**< **ActiveMQDestination** > **subscribedDestination**

6.659.1 Constructor & Destructor Documentation

- 6.659.1.1** `activemq::commands::SubscriptionInfo::SubscriptionInfo (const SubscriptionInfo &) [inline, protected]`
- 6.659.1.2** `activemq::commands::SubscriptionInfo::SubscriptionInfo ()`
- 6.659.1.3** `virtual activemq::commands::SubscriptionInfo::~~SubscriptionInfo () [virtual]`

6.659.2 Member Function Documentation

- 6.659.2.1** `virtual SubscriptionInfo* activemq::commands::SubscriptionInfo::cloneDataStructure () const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1373).

- 6.659.2.2** `virtual void activemq::commands::SubscriptionInfo::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

Implements `activemq::commands::DataStructure` (p. 1374).

- 6.659.2.3** `virtual bool activemq::commands::SubscriptionInfo::equals (const DataStructure * value) const [virtual]`

Compares the `DataStructure` (p. 1372) passed in to this one, and returns if they are equivalent.

Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Implements `activemq::commands::DataStructure` (p. 1374).

6.659.2.4 virtual std::string& activemq::commands::SubscriptionInfo::getClientId () [virtual]

6.659.2.5 virtual const std::string& activemq::commands::SubscriptionInfo::getClientId () const [virtual]

6.659.2.6 virtual unsigned char activemq::commands::SubscriptionInfo::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataSet** (p. 1372) type copy.

Implements **activemq::commands::DataSet** (p. 1375).

- 6.659.2.7 `virtual const Pointer<ActiveMQDestination>& activemq::commands::SubscriptionInfo::getDestination () const` [virtual]
- 6.659.2.8 `virtual Pointer<ActiveMQDestination>& activemq::commands::SubscriptionInfo::getDestination ()` [virtual]
- 6.659.2.9 `virtual const std::string& activemq::commands::SubscriptionInfo::getSelector () const` [virtual]
- 6.659.2.10 `virtual std::string& activemq::commands::SubscriptionInfo::getSelector ()` [virtual]
- 6.659.2.11 `virtual const std::string& activemq::commands::SubscriptionInfo::getSubscriptionName () const` [virtual]
- 6.659.2.12 `virtual std::string& activemq::commands::SubscriptionInfo::getSubscriptionName ()` [virtual]
- 6.659.2.13 `virtual const Pointer<ActiveMQDestination>& activemq::commands::SubscriptionInfo::getSubscribedDestination () const` [virtual]
- 6.659.2.14 `virtual Pointer<ActiveMQDestination>& activemq::commands::SubscriptionInfo::getSubscribedDestination ()` [virtual]
- 6.659.2.15 `SubscriptionInfo& activemq::commands::SubscriptionInfo::operator= (const SubscriptionInfo &)` [inline, protected]
- 6.659.2.16 `virtual void activemq::commands::SubscriptionInfo::setClientId (const std::string & clientId)` [virtual]
- 6.659.2.17 `virtual void activemq::commands::SubscriptionInfo::setDestination (const Pointer< ActiveMQDestination > & destination)` [virtual]
- 6.659.2.18 `virtual void activemq::commands::SubscriptionInfo::setSelector (const std::string & selector)` [virtual]
- 6.659.2.19 `virtual void activemq::commands::SubscriptionInfo::setSubscriptionName (const std::string & subscriptionName)` [virtual]
- 6.659.2.20 `virtual void activemq::commands::SubscriptionInfo::setSubscribedDestination (const Pointer< ActiveMQDestination > & subscribedDestination)` [virtual]
- 6.659.2.21 `virtual std::string activemq::commands::SubscriptionInfo::toString () const` [virtual]

Returns a string containing the information for this **DataStructure** (p.1372) such as its type and value of its elements. Generated on Mon Jan 24 2011 16:15:22 for activemq-cpp-3.1.0 by Doxygen

Returns

formatted string useful for debugging.

Reimplemented from `activemq::commands::BaseDataStructure` (p. 662).

6.659.3 Field Documentation

6.659.3.1 `std::string activemq::commands::SubscriptionInfo::clientId` [protected]

6.659.3.2 `Pointer<ActiveMQDestination> activemq::commands::SubscriptionInfo::destination`
[protected]

6.659.3.3 `const unsigned char activemq::commands::SubscriptionInfo::ID_ - SUBSCRIPTIONINFO = 55` [static]

6.659.3.4 `std::string activemq::commands::SubscriptionInfo::selector` [protected]

6.659.3.5 `std::string activemq::commands::SubscriptionInfo::subscriptionName`
[protected]

6.659.3.6 `Pointer<ActiveMQDestination> activemq::commands::SubscriptionInfo::subscribedDestination`
[protected]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/SubscriptionInfo.h`

6.660 activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller

Class Reference

Marshaling code for Open Wire Format for `SubscriptionInfoMarshaller` (p. 2911).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/SubscriptionInfoMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller`:

Public Member Functions

- `SubscriptionInfoMarshaller ()`
- `virtual ~SubscriptionInfoMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaller.

- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.660.1 Detailed Description

Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 2911). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.660.2 Constructor & Destructor Documentation

6.660.2.1 activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller::SubscriptionInfoMarshaller () [inline]

6.660.2.2 virtual activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller::~~SubscriptionInfoMarshaller () [inline, virtual]

6.660.3 Member Function Documentation

6.660.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1331).

6.660.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller::getDataStructureType() const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1336).

6.660.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1342).

6.660.3.4 `virtual void activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller::looseUnmarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1348).

```

6.660.3.5 virtual int ac-
tivemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, utils::BooleanStream * bs ) throw (
decaf::io::IOException ) [virtual]

```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1354).

```

6.660.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1360).

6.660.3.7 virtual void activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1366).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/SubscriptionInfoMarshaller.h

6.661 activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 2915).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/SubscriptionInfoMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller:

Public Member Functions

- **SubscriptionInfoMarshaller** ()
- virtual **~SubscriptionInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

6.661.1 Detailed Description

Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 2915). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.661.2 Constructor & Destructor Documentation

6.661.2.1 **activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller::SubscriptionInfoMarshaller** () [inline]

6.661.2.2 **virtual** **activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller::~~SubscriptionInfoMarshaller** () [inline, virtual]

6.661.3 Member Function Documentation

6.661.3.1 **virtual commands::DataStructure*** **activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller::createObject** () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1331).

6.661.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller::getDataStructureType() const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1336).

6.661.3.3 virtual void activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller::looseMarshal(OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1342).

6.661.3.4 virtual void activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller::looseUnmarshal(OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1348).

```

6.661.3.5  virtual int ac-
            tivemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller::tightMarshal1
            ( OpenWireFormat * wireFormat, commands::DataStructure
            * dataStructure, utils::BooleanStream * bs ) throw (
            decaf::io::IOException ) [virtual]

```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1354).

```

6.661.3.6  virtual void ac-
            tivemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller::tightMarshal2
            ( OpenWireFormat * wireFormat, commands::DataStructure
            * dataStructure, decaf::io::DataOutputStream * dataOut,
            utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1360).

```

6.661.3.7  virtual void ac-
            tivemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller::tightUnmarshal
            ( OpenWireFormat * wireFormat, commands::DataStructure
            * dataStructure, decaf::io::DataInputStream * dataIn,
            utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]

```

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1366).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/SubscriptionInfoMarshaller.h`

6.662 activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 2919).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/SubscriptionInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller**:

Public Member Functions

- **SubscriptionInfoMarshaller** ()
- virtual **~SubscriptionInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

- virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.662.1 Detailed Description

Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 2919). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.662.2 Constructor & Destructor Documentation

6.662.2.1 activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller::SubscriptionInfoMarshaller () [inline]

6.662.2.2 virtual activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller::~~SubscriptionInfoMarshaller () [inline, virtual]

6.662.3 Member Function Documentation

6.662.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1331).

6.662.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1336).

6.662.3.3 virtual void activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1342).

6.662.3.4 virtual void activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1348).

6.662.3.5 virtual int activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1354).

```
6.662.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1360).

```
6.662.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1366).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/SubscriptionInfoMarshaller.h`

6.663 **activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller** Class Reference

Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 2923).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/SubscriptionInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller**:

Public Member Functions

- **SubscriptionInfoMarshaller** ()
- virtual **~SubscriptionInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

6.663.1 Detailed Description

Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 2923). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.663.2 Constructor & Destructor Documentation

6.663.2.1 `activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller::SubscriptionInfoMarshaller () [inline]`

6.663.2.2 `virtual activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller::~~SubscriptionInfoMarshaller () [inline, virtual]`

6.663.3 Member Function Documentation

6.663.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.663.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.663.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1342).

6.663.3.4 virtual void activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1348).

6.663.3.5 virtual int activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1354).

6.663.3.6 virtual void activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1360).

6.663.3.7 virtual void activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1366).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/SubscriptionInfoMarshaller.h`

6.664 activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for `SubscriptionInfoMarshaller` (p. 2926).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/SubscriptionInfoMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller:

Public Member Functions

- **SubscriptionInfoMarshaller** ()
- virtual **~SubscriptionInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.664.1 Detailed Description

Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 2926). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.664.2 Constructor & Destructor Documentation

6.664.2.1 `activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller::SubscriptionInfoMarshaller()` [inline]

6.664.2.2 `virtual activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller::~~SubscriptionInfoMarshaller()` [inline, virtual]

6.664.3 Member Function Documentation

6.664.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.664.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.664.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller::looseMarshal(OpenWireFormat* wireFormat, commands::DataStructure* dataStructure, decaf::io::DataOutputStream* dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1342).

6.664.3.4 virtual void activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1348).

6.664.3.5 virtual int activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1354).

6.664.3.6 virtual void activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1360).

6.664.3.7 virtual void **activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller::tightUnmarshal**
 (**OpenWireFormat** * *wireFormat*, **commands::DataStructure**
 * *dataStructure*, **decaf::io::DataInputStream** * *dataIn*,
utils::BooleanStream * *bs*) throw (**decaf::io::IOException**) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1366).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/SubscriptionInfoMarshaller.h`

6.665 decaf::util::concurrent::Synchronizable Class Reference

The interface for all synchronizable objects (that is, objects that can be locked and unlocked).

```
#include <src/main/decaf/util/concurrent/Synchronizable.h>
```

Inheritance diagram for `decaf::util::concurrent::Synchronizable`:

Public Member Functions

- virtual \sim **Synchronizable** ()
- virtual void **lock** ()=0 throw (decaf::lang::exceptions::RuntimeException)
Locks the object.
- virtual bool **tryLock** ()=0 throw (decaf::lang::exceptions::RuntimeException)
*Attempts to **Lock** (p.1903) the object, if the lock is already held by another thread than this method returns false.*
- virtual void **unlock** ()=0 throw (decaf::lang::exceptions::RuntimeException)
Unlocks the object.
- virtual void **wait** ()=0 throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (long long millisecs)=0 throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (long long millisecs, int nanos)=0 throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **notify** ()=0 throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)
Signals a waiter on this object that it can now wake up and continue.
- virtual void **notifyAll** ()=0 throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)
Signals the waiters on this object that it can now wake up and continue.

6.665.1 Detailed Description

The interface for all synchronizable objects (that is, objects that can be locked and unlocked).

Since

1.0

6.665.2 Constructor & Destructor Documentation

6.665.2.1 virtual decaf::util::concurrent::Synchronizable::~~Synchronizable ()
[inline, virtual]

6.665.3 Member Function Documentation

6.665.3.1 virtual void decaf::util::concurrent::Synchronizable::lock () throw (decaf::lang::exceptions::RuntimeException) [pure virtual]

Locks the object.

Exceptions

RuntimeException if an error occurs while locking the object.

Implemented in `activemq::core::MessageDispatchChannel` (p. 2092), `decaf::internal::io::StandardErrorOutputStream` (p. 2814), `decaf::internal::io::StandardInputStream` (p. 2820), `decaf::internal::io::StandardOutputStream` (p. 2827), `decaf::internal::util::concurrent::SynchronizableImpl` (p. 2943), `decaf::io::BlockingByteArrayInputStream` (p. 669), `decaf::io::ByteArrayInputStream` (p. 831), `decaf::io::ByteArrayOutputStream` (p. 838), `decaf::io::FilterInputStream` (p. 1531), `decaf::io::FilterOutputStream` (p. 1539), `decaf::net::SocketInputStream` (p. 2799), `decaf::net::SocketOutputStream` (p. 2805), `decaf::util::AbstractCollection< E >` (p. 126), `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >` (p. 1033), `decaf::util::concurrent::Mutex` (p. 2240), `decaf::util::StlMap< K, V, COMPARATOR >` (p. 2853), `decaf::util::StlQueue< T >` (p. 2862), `decaf::util::AbstractCollection< transport::TransportListener * >` (p. 126), `decaf::util::AbstractCollection< Pointer< Synchronization > >` (p. 126), `decaf::util::AbstractCollection< CompositeTask * >` (p. 126), `decaf::util::AbstractCollection< URI >` (p. 126), `decaf::util::AbstractCollection< ActiveMQSession * >` (p. 126), `decaf::util::AbstractCollection< Pointer< DestinationInfo > >` (p. 126), `decaf::util::AbstractCollection< PrimitiveValueNode >` (p. 126), `decaf::util::AbstractCollection< Pointer< Command > >` (p. 126), `decaf::util::AbstractCollection< Pointer< BackupTransport > >` (p. 126), `decaf::util::concurrent::ConcurrentStlMap< Pointer< TransactionId >, Pointer< TransactionState >, TransactionId::COMPARATOR >` (p. 1033), `decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >` (p. 1033), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >` (p. 1033), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >` (p. 1033), `decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >` (p. 1033), `decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >` (p. 1033), `decaf::util::StlMap< cms::Session *, SessionResolver * >` (p. 2853), `decaf::util::StlMap< std::string, WireFormatFactory * >` (p. 2853), `decaf::util::StlMap< std::string, PrimitiveValueNode >` (p. 2853), `decaf::util::StlMap< std::string, cms::Queue * >` (p. 2853), `decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR >` (p. 2853), `decaf::util::StlMap< std::string, CachedConsumer * >` (p. 2853), `decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >` (p. 2853),

decaf::util::StlMap< std::string, TransportFactory * > (p. 2853), decaf::util::StlMap< int, Pointer< Command > > (p. 2853), decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR > (p. 2853), decaf::util::StlMap< std::string, CachedProducer * > (p. 2853), decaf::util::StlMap< std::string, cms::Topic * > (p. 2853), decaf::util::StlQueue< Pointer< Transport > > (p. 2862), decaf::util::StlQueue< Pointer< MessageDispatch > > (p. 2862), decaf::util::StlQueue< Task > (p. 2862), decaf::util::StlQueue< Pointer< Command > > (p. 2862), and decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > > (p. 2862).

Referenced by decaf::util::concurrent::Lock::lock().

6.665.3.2 virtual void decaf::util::concurrent::Synchronizable::notify
() throw (decaf::lang::exceptions::RuntimeException,
decaf::lang::exceptions::IllegalMonitorStateException) [pure virtual]

Signals a waiter on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

IllegalMonitorStateException - if the current thread is not the owner of the the **Synchronizable** (p. 2930) Object.

RuntimeException if an error occurs while notifying one of the waiting threads.

Implemented in **activemq::core::MessageDispatchChannel** (p. 2092),
decaf::internal::io::StandardErrorOutputStream (p. 2814), **decaf::internal::io::StandardInputStream** (p. 2821), **decaf::internal::io::StandardOutputStream** (p. 2827), **decaf::internal::util::concurrent::SynchronizableImpl** (p. 2943), **decaf::io::BlockingByteArrayInputStream** (p. 669), **decaf::io::ByteArrayInputStream** (p. 831), **decaf::io::ByteArrayOutputStream** (p. 838), **decaf::io::FilterInputStream** (p. 1532), **decaf::io::FilterOutputStream** (p. 1540), **decaf::net::SocketInputStream** (p. 2799), **decaf::net::SocketOutputStream** (p. 2806), **decaf::util::AbstractCollection< E >** (p. 126), **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 1033), **decaf::util::concurrent::Mutex** (p. 2241), **decaf::util::StlMap< K, V, COMPARATOR >** (p. 2853), **decaf::util::StlQueue< T >** (p. 2862), **decaf::util::AbstractCollection< transport::TransportListener * >** (p. 126), **decaf::util::AbstractCollection< Pointer< Synchronization > >** (p. 126), **decaf::util::AbstractCollection< CompositeTask * >** (p. 126), **decaf::util::AbstractCollection< URI >** (p. 126), **decaf::util::AbstractCollection< ActiveMQSession * >** (p. 126), **decaf::util::AbstractCollection< Pointer< DestinationInfo > >** (p. 126), **decaf::util::AbstractCollection< PrimitiveValueNode >** (p. 126), **decaf::util::AbstractCollection< Pointer< Command > >** (p. 126), **decaf::util::AbstractCollection< Pointer< BackupTransport > >** (p. 126), **decaf::util::concurrent::ConcurrentStlMap< Pointer< TransactionId >, Pointer< TransactionState >, TransactionId::COMPARATOR >** (p. 1033), **decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >** (p. 1033), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >** (p. 1033), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >** (p. 1033), **decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId**

>, Pointer< SessionState >, SessionId::COMPARATOR > (p. 1033), decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR > (p. 1033), decaf::util::StlMap< cms::Session *, SessionResolver * > (p. 2853), decaf::util::StlMap< std::string, WireFormatFactory * > (p. 2853), decaf::util::StlMap< std::string, PrimitiveValueNode > (p. 2853), decaf::util::StlMap< std::string, cms::Queue * > (p. 2853), decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR > (p. 2853), decaf::util::StlMap< std::string, CachedConsumer * > (p. 2853), decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR > (p. 2853), decaf::util::StlMap< std::string, TransportFactory * > (p. 2853), decaf::util::StlMap< int, Pointer< Command > > (p. 2853), decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR > (p. 2853), decaf::util::StlMap< std::string, CachedProducer * > (p. 2853), decaf::util::StlMap< std::string, cms::Topic * > (p. 2853), decaf::util::StlQueue< Pointer< Transport > > (p. 2862), decaf::util::StlQueue< Pointer< MessageDispatch > > (p. 2862), decaf::util::StlQueue< Task > (p. 2862), decaf::util::StlQueue< Pointer< Command > > (p. 2862), and decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > > (p. 2862).

6.665.3.3 virtual void decaf::util::concurrent::Synchronizable::notifyAll
 () throw (decaf::lang::exceptions::RuntimeException,
 decaf::lang::exceptions::IllegalMonitorStateException) [pure virtual]

Signals the waiters on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

IllegalMonitorStateException - if the current thread is not the owner of the the **Synchronizable** (p. 2930) Object.

RuntimeException if an error occurs while notifying the waiting threads.

Implemented in **activemq::core::MessageDispatchChannel** (p. 2092), **decaf::internal::io::StandardErrorOutputStream** (p. 2815), **decaf::internal::io::StandardInputStream** (p. 2821), **decaf::internal::io::StandardOutputStream** (p. 2827), **decaf::internal::util::concurrent::SynchronizableImpl** (p. 2943), **decaf::io::BlockingByteArrayInputStream** (p. 670), **decaf::io::ByteArrayInputStream** (p. 832), **decaf::io::ByteArrayOutputStream** (p. 839), **decaf::io::FilterInputStream** (p. 1532), **decaf::io::FilterOutputStream** (p. 1540), **decaf::net::SocketInputStream** (p. 2800), **decaf::net::SocketOutputStream** (p. 2806), **decaf::util::AbstractCollection< E >** (p. 126), **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 1033), **decaf::util::concurrent::Mutex** (p. 2241), **decaf::util::StlMap< K, V, COMPARATOR >** (p. 2853), **decaf::util::StlQueue< T >** (p. 2862), **decaf::util::AbstractCollection< transport::TransportListener * >** (p. 126), **decaf::util::AbstractCollection< Pointer< Synchronization > >** (p. 126), **decaf::util::AbstractCollection< CompositeTask * >** (p. 126), **decaf::util::AbstractCollection< URI >** (p. 126), **decaf::util::AbstractCollection< ActiveMQSession * >** (p. 126), **decaf::util::AbstractCollection< Pointer< DestinationInfo > >** (p. 126), **decaf::util::AbstractCollection< PrimitiveValueNode >** (p. 126), **decaf::util::AbstractCollection< Pointer< Command > >** (p. 126), **decaf::util::AbstractCollection< Pointer< BackupTransport > >** (p. 126), **decaf::util::concurrent::ConcurrentStlMap<**

Pointer< TransactionId >, Pointer< TransactionState >, TransactionId::COMPARATOR > (p. 1033), decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR > (p. 1033), decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR > (p. 1033), decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR > (p. 1033), decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR > (p. 1033), decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR > (p. 1033), decaf::util::StlMap< cms::Session *, SessionResolver * > (p. 2853), decaf::util::StlMap< std::string, WireFormatFactory * > (p. 2853), decaf::util::StlMap< std::string, PrimitiveValueNode > (p. 2853), decaf::util::StlMap< std::string, cms::Queue * > (p. 2853), decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR > (p. 2853), decaf::util::StlMap< std::string, CachedConsumer * > (p. 2853), decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR > (p. 2853), decaf::util::StlMap< std::string, TransportFactory * > (p. 2853), decaf::util::StlMap< int, Pointer< Command > > (p. 2853), decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR > (p. 2853), decaf::util::StlMap< std::string, CachedProducer * > (p. 2853), decaf::util::StlMap< std::string, cms::Topic * > (p. 2853), decaf::util::StlQueue< Pointer< Transport > > (p. 2862), decaf::util::StlQueue< Pointer< MessageDispatch > > (p. 2862), decaf::util::StlQueue< Task > (p. 2862), decaf::util::StlQueue< Pointer< Command > > (p. 2862), and decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > > (p. 2862).

6.665.3.4 virtual bool decaf::util::concurrent::Synchronizable::tryLock () throw (decaf::lang::exceptions::RuntimeException) [pure virtual]

Attempts to **Lock** (p. 1903) the object, if the lock is already held by another thread than this method returns false.

Returns

true if the lock was acquired, false if it is already held by another thread.

Exceptions

RuntimeException if an error occurs while locking the object.

Implemented in `activemq::core::MessageDispatchChannel` (p. 2093), `decaf::internal::io::StandardOutputStream` (p. 2815), `decaf::internal::io::StandardInputStream` (p. 2823), `decaf::internal::io::StandardOutputStream` (p. 2828), `decaf::internal::util::concurrent::SynchronizableImpl` (p. 2944), `decaf::io::BlockingByteArrayInputStream` (p. 672), `decaf::io::ByteArrayInputStream` (p. 834), `decaf::io::ByteArrayOutputStream` (p. 840), `decaf::io::FilterInputStream` (p. 1534), `decaf::io::FilterOutputStream` (p. 1540), `decaf::net::SocketInputStream` (p. 2801), `decaf::net::SocketOutputStream` (p. 2806), `decaf::util::AbstractCollection< E >` (p. 129), `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >` (p. 1038), `decaf::util::concurrent::Mutex` (p. 2241), `decaf::util::StlMap< K, V, COMPARATOR >` (p. 2855), `decaf::util::StlQueue< T >` (p. 2863), `decaf::util::AbstractCollection< transport::TransportListener *`

> (p. 129), decaf::util::AbstractCollection< Pointer< Synchronization >
 > (p. 129), decaf::util::AbstractCollection< CompositeTask * > (p. 129),
 decaf::util::AbstractCollection< URI > (p. 129), decaf::util::AbstractCollection<
 ActiveMQSession * > (p. 129), decaf::util::AbstractCollection< Pointer<
 DestinationInfo > > (p. 129), decaf::util::AbstractCollection< Primitive-
 ValueNode > (p. 129), decaf::util::AbstractCollection< Pointer< Com-
 mand > > (p. 129), decaf::util::AbstractCollection< Pointer< Back-
 upTransport > > (p. 129), decaf::util::concurrent::ConcurrentStlMap<
 Pointer< TransactionId >, Pointer< TransactionState >, Transac-
 tionId::COMPARATOR > (p. 1038), decaf::util::concurrent::ConcurrentStlMap<
 Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR
 > (p. 1038), decaf::util::concurrent::ConcurrentStlMap< Pointer< Connec-
 tionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR
 > (p. 1038), decaf::util::concurrent::ConcurrentStlMap< Pointer< Con-
 sumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR
 > (p. 1038), decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId
 >, Pointer< SessionState >, SessionId::COMPARATOR > (p. 1038),
 decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer<
 ProducerState >, ProducerId::COMPARATOR > (p. 1038), decaf::util::StlMap<
 cms::Session *, SessionResolver * > (p. 2855), decaf::util::StlMap< std::string,
 WireFormatFactory * > (p. 2855), decaf::util::StlMap< std::string, PrimitiveVal-
 ueNode > (p. 2855), decaf::util::StlMap< std::string, cms::Queue * > (p. 2855),
 decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, com-
 mands::ProducerId::COMPARATOR > (p. 2855), decaf::util::StlMap< std::string,
 CachedConsumer * > (p. 2855), decaf::util::StlMap< Pointer< commands::ConsumerId
 >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR > (p. 2855),
 decaf::util::StlMap< std::string, TransportFactory * > (p. 2855), decaf::util::StlMap<
 int, Pointer< Command > > (p. 2855), decaf::util::StlMap< Pointer< com-
 mands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR
 > (p. 2855), decaf::util::StlMap< std::string, CachedProducer * > (p. 2855),
 decaf::util::StlMap< std::string, cms::Topic * > (p. 2855), decaf::util::StlQueue<
 Pointer< Transport > > (p. 2863), decaf::util::StlQueue< Pointer< MessageDis-
 patch > > (p. 2863), decaf::util::StlQueue< Task > (p. 2863), decaf::util::StlQueue<
 Pointer< Command > > (p. 2863), and decaf::util::StlQueue< decaf::lang::Pointer<
 commands::MessageDispatch > > (p. 2863).

6.665.3.5 virtual void decaf::util::concurrent::Synchronizable::unlock () throw (decaf::lang::exceptions::RuntimeException) [pure virtual]

Unlocks the object.

Exceptions

RuntimeException if an error occurs while unlocking the object.

Implemented in activemq::core::MessageDispatchChannel (p. 2094), decaf::internal::io::StandardOutputStream (p. 2815), decaf::internal::io::StandardInputStream (p. 2823), decaf::internal::io::StandardOutputStream (p. 2828), decaf::internal::util::concurrent::SynchronizableImpl (p. 2944), decaf::io::BlockingByteArrayInputStream (p. 672), decaf::io::ByteArrayInputStream (p. 834), decaf::io::ByteArrayOutputStream (p. 841), decaf::io::FilterInputStream (p. 1535), decaf::io::FilterOutputStream (p. 1540), decaf::net::SocketInputStream (p. 2802), decaf::net::SocketOutputStream (p. 2806), decaf::util::AbstractCollection<

E > (p. 129), decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR > (p. 1038), decaf::util::concurrent::Mutex (p. 2242), decaf::util::StlMap< K, V, COMPARATOR > (p. 2856), decaf::util::StlQueue< T > (p. 2864), decaf::util::AbstractCollection< transport::TransportListener * > (p. 129), decaf::util::AbstractCollection< Pointer< Synchronization > > (p. 129), decaf::util::AbstractCollection< CompositeTask * > (p. 129), decaf::util::AbstractCollection< URI > (p. 129), decaf::util::AbstractCollection< ActiveMQSession * > (p. 129), decaf::util::AbstractCollection< Pointer< DestinationInfo > > (p. 129), decaf::util::AbstractCollection< PrimitiveValueNode > > (p. 129), decaf::util::AbstractCollection< Pointer< Command > > (p. 129), decaf::util::AbstractCollection< Pointer< BackupTransport > > (p. 129), decaf::util::concurrent::ConcurrentStlMap< Pointer< TransactionId >, Pointer< TransactionState >, TransactionId::COMPARATOR > (p. 1038), decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR > (p. 1038), decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR > (p. 1038), decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR > (p. 1038), decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR > (p. 1038), decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR > (p. 1038), decaf::util::StlMap< cms::Session *, SessionResolver * > (p. 2856), decaf::util::StlMap< std::string, WireFormatFactory * > (p. 2856), decaf::util::StlMap< std::string, PrimitiveValueNode > (p. 2856), decaf::util::StlMap< std::string, cms::Queue * > (p. 2856), decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR > (p. 2856), decaf::util::StlMap< std::string, CachedConsumer * > (p. 2856), decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR > (p. 2856), decaf::util::StlMap< std::string, TransportFactory * > (p. 2856), decaf::util::StlMap< int, Pointer< Command > > (p. 2856), decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR > (p. 2856), decaf::util::StlMap< std::string, CachedProducer * > (p. 2856), decaf::util::StlMap< std::string, cms::Topic * > (p. 2856), decaf::util::StlQueue< Pointer< Transport > > (p. 2864), decaf::util::StlQueue< Pointer< MessageDispatch > > (p. 2864), decaf::util::StlQueue< Task > (p. 2864), decaf::util::StlQueue< Pointer< Command > > (p. 2864), and decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > > (p. 2864).

Referenced by decaf::util::concurrent::Lock::unlock(), and decaf::util::concurrent::Lock::~~Lock().

6.665.3.6 virtual void decaf::util::concurrent::Synchronizable::wait () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException) [pure virtual]

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling.

Exceptions

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the *Synchronizable* (p. 2930) Object.

Implemented in *activemq::core::MessageDispatchChannel* (p. 2095), *decaf::internal::io::StandardErrorOutputStream* (p. 2815), *decaf::internal::io::StandardInputStream* (p. 2825), *decaf::internal::io::StandardOutputStream* (p. 2828), *decaf::internal::util::concurrent::SynchronizableImpl* (p. 2944), *decaf::io::BlockingByteArrayInputStream* (p. 672), *decaf::io::ByteArrayInputStream* (p. 835), *decaf::io::ByteArrayOutputStream* (p. 841), *decaf::io::FilterInputStream* (p. 1536), *decaf::io::FilterOutputStream* (p. 1541), *decaf::net::SocketInputStream* (p. 2803), *decaf::net::SocketOutputStream* (p. 2807), *decaf::util::AbstractCollection< E >* (p. 130), *decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >* (p. 1039), *decaf::util::concurrent::Mutex* (p. 2242), *decaf::util::StlMap< K, V, COMPARATOR >* (p. 2856), *decaf::util::StlQueue< T >* (p. 2864), *decaf::util::AbstractCollection< transport::TransportListener * >* (p. 130), *decaf::util::AbstractCollection< Pointer< Synchronization > >* (p. 130), *decaf::util::AbstractCollection< CompositeTask * >* (p. 130), *decaf::util::AbstractCollection< URI >* (p. 130), *decaf::util::AbstractCollection< ActiveMQSession * >* (p. 130), *decaf::util::AbstractCollection< Pointer< DestinationInfo > >* (p. 130), *decaf::util::AbstractCollection< PrimitiveValueNode >* (p. 130), *decaf::util::AbstractCollection< Pointer< Command > >* (p. 130), *decaf::util::AbstractCollection< Pointer< BackupTransport > >* (p. 130), *decaf::util::concurrent::ConcurrentStlMap< Pointer< TransactionId >, Pointer< TransactionState >, TransactionId::COMPARATOR >* (p. 1039), *decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >* (p. 1039), *decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >* (p. 1039), *decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >* (p. 1039), *decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >* (p. 1039), *decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >* (p. 1039), *decaf::util::StlMap< cms::Session *, SessionResolver * >* (p. 2856), *decaf::util::StlMap< std::string, WireFormatFactory * >* (p. 2856), *decaf::util::StlMap< std::string, PrimitiveValueNode >* (p. 2856), *decaf::util::StlMap< std::string, cms::Queue * >* (p. 2856), *decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR >* (p. 2856), *decaf::util::StlMap< std::string, CachedConsumer * >* (p. 2856), *decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >* (p. 2856), *decaf::util::StlMap< std::string, TransportFactory * >* (p. 2856), *decaf::util::StlMap< int, Pointer< Command > >* (p. 2856), *decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR >* (p. 2856), *decaf::util::StlMap< std::string, CachedProducer * >* (p. 2856), *decaf::util::StlMap< std::string, cms::Topic * >* (p. 2856), *decaf::util::StlQueue< Pointer< Transport > >* (p. 2864), *decaf::util::StlQueue< Pointer< MessageDispatch > >* (p. 2864), *decaf::util::StlQueue< Task >* (p. 2864), *decaf::util::StlQueue< Pointer< Command > >* (p. 2864), and *decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >* (p. 2864).

6.665.3.7 virtual void decaf::util::concurrent::Synchronizable::wait (long long *milliseconds*) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException) [pure virtual]

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval.

Parameters

milliseconds the time in milliseconds to wait, or WAIT_INFINITE

Exceptions

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the **Synchronizable** (p. 2930) Object.

Implemented in **activemq::core::MessageDispatchChannel** (p. 2094), **decaf::internal::io::StandardErrorOutputStream** (p. 2816), **decaf::internal::io::StandardInputStream** (p. 2824), **decaf::internal::io::StandardOutputStream** (p. 2828), **decaf::internal::util::concurrent::SynchronizableImpl** (p. 2944), **decaf::io::BlockingByteArrayInputStream** (p. 673), **decaf::io::ByteArrayInputStream** (p. 835), **decaf::io::ByteArrayOutputStream** (p. 841), **decaf::io::FilterInputStream** (p. 1535), **decaf::io::FilterOutputStream** (p. 1541), **decaf::net::SocketInputStream** (p. 2802), **decaf::net::SocketOutputStream** (p. 2807), **decaf::util::AbstractCollection< E >** (p. 131), **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 1038), **decaf::util::concurrent::Mutex** (p. 2243), **decaf::util::StlMap< K, V, COMPARATOR >** (p. 2857), **decaf::util::StlQueue< T >** (p. 2864), **decaf::util::AbstractCollection< transport::TransportListener * >** (p. 131), **decaf::util::AbstractCollection< Pointer< Synchronization > >** (p. 131), **decaf::util::AbstractCollection< CompositeTask * >** (p. 131), **decaf::util::AbstractCollection< URI >** (p. 131), **decaf::util::AbstractCollection< ActiveMQSession * >** (p. 131), **decaf::util::AbstractCollection< Pointer< DestinationInfo > >** (p. 131), **decaf::util::AbstractCollection< PrimitiveValueNode >** (p. 131), **decaf::util::AbstractCollection< Pointer< Command > >** (p. 131), **decaf::util::AbstractCollection< Pointer< BackupTransport > >** (p. 131), **decaf::util::concurrent::ConcurrentStlMap< Pointer< TransactionId >, Pointer< TransactionState >, TransactionId::COMPARATOR >** (p. 1038), **decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >** (p. 1038), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >** (p. 1038), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >** (p. 1038), **decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >** (p. 1038), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >** (p. 1038), **decaf::util::StlMap< cms::Session *, SessionResolver * >** (p. 2857), **decaf::util::StlMap< std::string, WireFormatFactory * >** (p. 2857), **decaf::util::StlMap< std::string, PrimitiveValueNode >** (p. 2857), **decaf::util::StlMap< std::string, cms::Queue * >** (p. 2857),

`decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR > (p. 2857), decaf::util::StlMap< std::string, CachedConsumer * > (p. 2857), decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR > (p. 2857), decaf::util::StlMap< std::string, TransportFactory * > (p. 2857), decaf::util::StlMap< int, Pointer< Command > > (p. 2857), decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR > (p. 2857), decaf::util::StlMap< std::string, CachedProducer * > (p. 2857), decaf::util::StlMap< std::string, cms::Topic * > (p. 2857), decaf::util::StlQueue< Pointer< Transport > > (p. 2864), decaf::util::StlQueue< Pointer< MessageDispatch > > (p. 2864), decaf::util::StlQueue< Task > (p. 2864), decaf::util::StlQueue< Pointer< Command > > (p. 2864), and decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > > (p. 2864).`

6.665.3.8 `virtual void decaf::util::concurrent::Synchronizable::wait (long long millisecs, int nanos) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException) [pure virtual]`

Waits on a signal from this object, which is generated by a call to `Notify`.

Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

Parameters

millisecs the time in milliseconds to wait, or `WAIT_INFINITE`

nanos additional time in nanoseconds with a range of 0-999999

Exceptions

IllegalArgumentException if an error occurs or the *nanos* argument is not in the range of [0-999999]

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the **Synchronizable** (p. 2930) Object.

Implemented in `activemq::core::MessageDispatchChannel` (p. 2094), `decaf::internal::io::StandardErrorOutputStream` (p. 2816), `decaf::internal::io::StandardInputStream` (p. 2824), `decaf::internal::io::StandardOutputStream` (p. 2829), `decaf::internal::util::concurrent::SynchronizableImpl` (p. 2945), `decaf::io::BlockingByteArrayInputStream` (p. 673), `decaf::io::ByteArrayInputStream` (p. 834), `decaf::io::ByteArrayOutputStream` (p. 842), `decaf::io::FilterInputStream` (p. 1535), `decaf::io::FilterOutputStream` (p. 1541), `decaf::net::SocketInputStream` (p. 2802), `decaf::net::SocketOutputStream` (p. 2807), `decaf::util::AbstractCollection< E >` (p. 130), `decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >` (p. 1039), `decaf::util::concurrent::Mutex` (p. 2243), `decaf::util::StlMap<`

K, V, COMPARATOR > (p. 2856), decaf::util::StlQueue< T > (p. 2865), decaf::util::AbstractCollection< transport::TransportListener * > (p. 130), decaf::util::AbstractCollection< Pointer< Synchronization > > (p. 130), decaf::util::AbstractCollection< CompositeTask * > (p. 130), decaf::util::AbstractCollection< URI > (p. 130), decaf::util::AbstractCollection< ActiveMQSession * > (p. 130), decaf::util::AbstractCollection< Pointer< DestinationInfo > > (p. 130), decaf::util::AbstractCollection< PrimitiveValueNode > > (p. 130), decaf::util::AbstractCollection< Pointer< Command > > (p. 130), decaf::util::AbstractCollection< Pointer< BackupTransport > > (p. 130), decaf::util::concurrent::ConcurrentStlMap< Pointer< TransactionId >, Pointer< TransactionState >, TransactionId::COMPARATOR > (p. 1039), decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR > (p. 1039), decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR > (p. 1039), decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR > (p. 1039), decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR > (p. 1039), decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR > (p. 1039), decaf::util::StlMap< cms::Session *, SessionResolver * > (p. 2856), decaf::util::StlMap< std::string, WireFormatFactory * > (p. 2856), decaf::util::StlMap< std::string, PrimitiveValueNode > (p. 2856), decaf::util::StlMap< std::string, cms::Queue * > (p. 2856), decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR > (p. 2856), decaf::util::StlMap< std::string, CachedConsumer * > (p. 2856), decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR > (p. 2856), decaf::util::StlMap< std::string, TransportFactory * > (p. 2856), decaf::util::StlMap< int, Pointer< Command > > (p. 2856), decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR > (p. 2856), decaf::util::StlMap< std::string, CachedProducer * > (p. 2856), decaf::util::StlMap< std::string, cms::Topic * > (p. 2856), decaf::util::StlQueue< Pointer< Transport > > (p. 2865), decaf::util::StlQueue< Pointer< MessageDispatch > > (p. 2865), decaf::util::StlQueue< Task > (p. 2865), decaf::util::StlQueue< Pointer< Command > > (p. 2865), and decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > > (p. 2865).

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/Synchronizable.h

6.666 decaf::internal::util::concurrent::SynchronizableImpl Class Reference

A convenience class used by some Decaf classes to implement the Synchronizable interface when there is no issues related to multiple inheritance.

```
#include <src/main/decaf/internal/util/concurrent/SynchronizableImpl.h>
```

Inheritance diagram for decaf::internal::util::concurrent::SynchronizableImpl:

Public Member Functions

- **SynchronizableImpl** ()
- virtual \sim **SynchronizableImpl** ()
- virtual void **lock** () throw (decaf::lang::exceptions::RuntimeException)
Locks the object.
- virtual bool **tryLock** () throw (decaf::lang::exceptions::RuntimeException)
Attempts to Lock the object, if the lock is already held by another thread than this method returns false.
- virtual void **unlock** () throw (decaf::lang::exceptions::RuntimeException)
Unlocks the object.
- virtual void **wait** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (long long millisecs) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (long long millisecs, int nanos) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException)
Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **notify** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)
Signals a waiter on this object that it can now wake up and continue.
- virtual void **notifyAll** () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException)
Signals the waiters on this object that it can now wake up and continue.

6.666.1 Detailed Description

A convenience class used by some Decaf classes to implement the Synchronizable interface when there is no issues related to multiple inheritance.

Since

1.0

6.666.2 Constructor & Destructor Documentation

6.666.2.1 decaf::internal::util::concurrent::SynchronizableImpl::SynchronizableImpl ()

6.666.2.2 virtual decaf::internal::util::concurrent::SynchronizableImpl::~~SynchronizableImpl () [virtual]

6.666.3 Member Function Documentation

6.666.3.1 virtual void decaf::internal::util::concurrent::SynchronizableImpl::lock () throw (decaf::lang::exceptions::RuntimeException) [virtual]

Locks the object.

Exceptions

RuntimeException if an error occurs while locking the object.

Implements decaf::util::concurrent::Synchronizable (p. 2932).

6.666.3.2 virtual void decaf::internal::util::concurrent::SynchronizableImpl::notify () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException) [virtual]

Signals a waiter on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

RuntimeException if an error occurs while notifying one of the waiting threads.

Implements decaf::util::concurrent::Synchronizable (p. 2933).

6.666.3.3 virtual void decaf::internal::util::concurrent::SynchronizableImpl::notifyAll () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException) [virtual]

Signals the waiters on this object that it can now wake up and continue.

Must have this object locked before calling.

Exceptions

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

RuntimeException if an error occurs while notifying the waiting threads.

Implements decaf::util::concurrent::Synchronizable (p. 2934).

6.666.3.4 `virtual bool decaf::internal::util::concurrent::SynchronizableImpl::tryLock () throw (decaf::lang::exceptions::RuntimeException) [virtual]`

Attempts to Lock the object, if the lock is already held by another thread than this method returns false.

Returns

true if the lock was acquired, false if it is already held by another thread.

Exceptions

RuntimeException if an error occurs while locking the object.

Implements `decaf::util::concurrent::Synchronizable` (p. 2935).

6.666.3.5 `virtual void decaf::internal::util::concurrent::SynchronizableImpl::unlock () throw (decaf::lang::exceptions::RuntimeException) [virtual]`

Unlocks the object.

Exceptions

RuntimeException if an error occurs while unlocking the object.

Implements `decaf::util::concurrent::Synchronizable` (p. 2936).

6.666.3.6 `virtual void decaf::internal::util::concurrent::SynchronizableImpl::wait () throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException) [virtual]`

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling.

Exceptions

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements `decaf::util::concurrent::Synchronizable` (p. 2937).

6.666.3.7 `virtual void decaf::internal::util::concurrent::SynchronizableImpl::wait (long long millisecs) throw (decaf::lang::exceptions::RuntimeException, decaf::lang::exceptions::IllegalMonitorStateException, decaf::lang::exceptions::InterruptedException) [virtual]`

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval.

Parameters

millisecs the time in milliseconds to wait, or WAIT_INFINITE

Exceptions

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2938).

```
6.666.3.8 virtual void decaf::internal::util::concurrent::SynchronizableImpl::wait
( long long millisecs, int nanos ) throw
( decaf::lang::exceptions::RuntimeException, de-
caf::lang::exceptions::IllegalArgumentException,
decaf::lang::exceptions::IllegalMonitorStateException,
decaf::lang::exceptions::InterruptedException ) [virtual]
```

Waits on a signal from this object, which is generated by a call to Notify.

Must have this object locked before calling. This wait will timeout after the specified time interval. This method is similar to the one argument wait function except that it add a finer grained control over the amount of time that it waits by adding in the additional nanosecond argument.

NOTE: The ability to wait accurately at a nanosecond scale depends on the platform and OS that the Decaf API is running on, some systems do not provide an accurate enough clock to provide this level of granularity.

Parameters

millisecs the time in milliseconds to wait, or WAIT_INFINITE

nanos additional time in nanoseconds with a range of 0-999999

Exceptions

IllegalArgumentException if an error occurs or the nanos argument is not in the range of [0-999999]

RuntimeException if an error occurs while waiting on the object.

InterruptedException if the wait is interrupted before it completes.

IllegalMonitorStateException - if the current thread is not the owner of the the Synchronizable Object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2940).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/util/concurrent/**SynchronizableImpl.h**

6.667 activemq::core::Synchronization Class Reference

Transacted Object **Synchronization** (p. 2945), used to sync the events of a Transaction with the items in the Transaction.

```
#include <src/main/activemq/core/Synchronization.h>
```

Public Member Functions

- virtual `~Synchronization()`
- virtual void `beforeEnd()` throw (exceptions::ActiveMQException)
- virtual void `afterCommit()` throw (exceptions::ActiveMQException)
- virtual void `afterRollback()` throw (exceptions::ActiveMQException)

6.667.1 Detailed Description

Transacted Object **Synchronization** (p. 2945), used to sync the events of a Transaction with the items in the Transaction.

6.667.2 Constructor & Destructor Documentation

6.667.2.1 virtual `activemq::core::Synchronization::~~Synchronization()` [inline, virtual]

6.667.3 Member Function Documentation

6.667.3.1 virtual void `activemq::core::Synchronization::afterCommit()` throw (exceptions::ActiveMQException) [pure virtual]

6.667.3.2 virtual void `activemq::core::Synchronization::afterRollback()` throw (exceptions::ActiveMQException) [pure virtual]

6.667.3.3 virtual void `activemq::core::Synchronization::beforeEnd()` throw (exceptions::ActiveMQException) [pure virtual]

The documentation for this class was generated from the following file:

- `src/main/activemq/core/Synchronization.h`

6.668 decaf::util::concurrent::SynchronousQueue< E > Class Template Reference

A `BlockingQueue` blocking queue} in which each insert operation must wait for a corresponding remove operation by another thread, and vice versa.

```
#include <src/main/decaf/util/concurrent/SynchronousQueue.h>
```

Data Structures

- class `EmptyIterator`

Public Member Functions

- `SynchronousQueue()`
- virtual `~SynchronousQueue()`

- virtual void **put** (const E &value) throw (decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException)
Adds the specified element to this queue, waiting if necessary for another thread to receive it.
- virtual bool **offer** (const E &e, long timeout, const TimeUnit &unit) throw (decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException)
Inserts the specified element into this queue, waiting if necessary up to the specified wait time for another thread to receive it.
- virtual bool **offer** (const E &value) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException)
Inserts the specified element into this queue, if another thread is waiting to receive it.
- virtual E **take** () throw (decaf::lang::exceptions::InterruptedException)
Retrieves and removes the head of this queue, waiting if necessary for another thread to insert it.
- virtual bool **poll** (E &result, long long timeout, const TimeUnit &unit) throw (decaf::lang::exceptions::InterruptedException)
Retrieves and removes the head of this queue, waiting if necessary up to the specified wait time, for another thread to insert it.
- virtual bool **poll** (E &result)
Retrieves and removes the head of this queue, if another thread is currently making an element available.
- virtual bool **equals** (const Collection< E > &value) const
- virtual decaf::util::Iterator< E > * **iterator** ()
- virtual decaf::util::Iterator< E > * **iterator** () const
- virtual bool **isEmpty** () const
- virtual std::size_t **size** () const
- virtual int **remainingCapacity** () const
- virtual void **clear** () throw (lang::exceptions::UnsupportedOperationException)
- virtual bool **contains** (const E &value DECAF_UNUSED) const throw (lang::Exception)
- virtual bool **containsAll** (const Collection< E > &collection) const throw (lang::Exception)
- virtual bool **remove** (const E &value DECAF_UNUSED) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException)
- virtual bool **removeAll** (const Collection< E > &collection DECAF_UNUSED) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException)
- virtual bool **retainAll** (const Collection< E > &collection DECAF_UNUSED) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException)
- virtual bool **peek** (E &result DECAF_UNUSED) const
- virtual std::vector< E > **toArray** () const
- virtual std::size_t **drainTo** (Collection< E > &c) throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IllegalArgumentException)

- virtual std::size_t **drainTo** (**Collection**< E > &c, std::size_t maxElements) throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IllegalArgumentException)

6.668.1 Detailed Description

```
template<typename E> class decaf::util::concurrent::SynchronousQueue< E >
```

A `BlockingQueue` blocking queue} in which each insert operation must wait for a corresponding remove operation by another thread, and vice versa. A synchronous queue does not have any internal capacity, not even a capacity of one. You cannot `peek` at a synchronous queue because an element is only present when you try to remove it; you cannot insert an element (using any method) unless another thread is trying to remove it; you cannot iterate as there is nothing to iterate. The *head* of the queue is the element that the first queued inserting thread is trying to add to the queue; if there is no such queued thread then no element is available for removal and `poll()` (p. 2951) will return `null`. For purposes of other `Collection` (p. 982) methods (for example `contains`), a `SynchronousQueue` (p. 2946) acts as an empty collection. This queue does not permit `null` elements.

Synchronous queues are similar to rendezvous channels used in CSP and Ada. They are well suited for handoff designs, in which an object running in one thread must sync up with an object running in another thread in order to hand it some information, event, or task.

This class supports an optional fairness policy for ordering waiting producer and consumer threads. By default, this ordering is not guaranteed. However, a queue constructed with fairness set to `true` grants threads access in FIFO order.

This class and its iterator implement all of the *optional* methods of the `Collection` (p. 982) and `Iterator` (p. 1716) interfaces.

Since

1.0

6.668.2 Constructor & Destructor Documentation

6.668.2.1 `template<typename E > decaf::util::concurrent::SynchronousQueue< E >::SynchronousQueue () [inline]`

6.668.2.2 `template<typename E > virtual decaf::util::concurrent::SynchronousQueue< E >::~~SynchronousQueue () [inline, virtual]`

6.668.3 Member Function Documentation

6.668.3.1 `template<typename E > virtual void decaf::util::concurrent::SynchronousQueue< E >::clear () throw (lang::exceptions::UnsupportedOperationException) [inline, virtual]`

6.668.3.2 `template<typename E > virtual bool decaf::util::concurrent::SynchronousQueue< E >::contains (const E &value DECAF_UNUSED) const throw (lang::Exception) [inline, virtual]`

6.668.3.3 `template<typename E > virtual bool decaf::util::concurrent::SynchronousQueue< E >::containsAll (const Collection< E > & collection) const throw (lang::Exception) [inline, virtual]`

6.668.3.4 `template<typename E > virtual std::size_t decaf::util::concurrent::SynchronousQueue< E >::drainTo (Collection< E > & c) throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IllegalArgumentException) [inline, virtual]`

References `decaf::util::concurrent::SynchronousQueue< E >::poll()`.

6.668.3.5 `template<typename E > virtual std::size_t decaf::util::concurrent::SynchronousQueue< E >::drainTo (Collection< E > & c, std::size_t maxElements) throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IllegalArgumentException) [inline, virtual]`

References `decaf::util::concurrent::SynchronousQueue< E >::poll()`.

- 6.668.3.6** `template<typename E > virtual bool
decaf::util::concurrent::SynchronousQueue< E >::equals (
const Collection< E > & value) const [inline, virtual]`
- 6.668.3.7** `template<typename E > virtual bool
decaf::util::concurrent::SynchronousQueue< E >::isEmpty
() const [inline, virtual]`
- 6.668.3.8** `template<typename E > virtual decaf::util::Iterator<E>*
decaf::util::concurrent::SynchronousQueue< E >::iterator () const
[inline, virtual]`
- 6.668.3.9** `template<typename E > virtual decaf::util::Iterator<E>*
decaf::util::concurrent::SynchronousQueue< E >::iterator () [inline,
virtual]`
- 6.668.3.10** `template<typename E > virtual bool
decaf::util::concurrent::SynchronousQueue< E >::offer (
const E & e, long timeout, const TimeUnit & unit
) throw (decaf::lang::exceptions::InterruptedException,
decaf::lang::exceptions::NullPointerException, de-
caf::lang::exceptions::IllegalArgumentException) [inline,
virtual]`

Inserts the specified element into this queue, waiting if necessary up to the specified wait time for another thread to receive it.

Returns

`true` if successful, or `false` if the specified waiting time elapses before a consumer appears.

Exceptions

InterruptedException

NullPointerException

IllegalArgumentException

- 6.668.3.11** `template<typename E > virtual bool
decaf::util::concurrent::SynchronousQueue< E >::offer (const E
& value) throw (decaf::lang::exceptions::NullPointerException,
decaf::lang::exceptions::IllegalArgumentException) [inline, virtual]`

Inserts the specified element into this queue, if another thread is waiting to receive it.

Parameters

value the element to add to the **Queue** (p.2507)

Returns

`true` if the element was added to this queue, else `false`

Exceptions

NullPointerException if the **Queue** (p. 2507) implementation does not allow Null values to be inserted into the **Queue** (p. 2507).

IllegalArgumentException if some property of the specified element prevents it from being added to this queue

6.668.3.12 `template<typename E > virtual bool
decaf::util::concurrent::SynchronousQueue< E >::peek (
E &result DECAF_UNUSED) const [inline, virtual]`

6.668.3.13 `template<typename E > virtual bool
decaf::util::concurrent::SynchronousQueue< E >::poll (
E & result, long long timeout, const TimeUnit & unit) throw (
decaf::lang::exceptions::InterruptedException) [inline, virtual]`

Retrieves and removes the head of this queue, waiting if necessary up to the specified wait time, for another thread to insert it.

Parameters

result a reference to the value where the head of the **Queue** (p. 2507) should be copied to.

timeout the time that the method should block if there is no element available to return.

unit the Time Units that the timeout value represents.

Returns

true if the head of the **Queue** (p. 2507) was copied to the result param or false if no value could be returned.

Referenced by decaf::util::concurrent::SynchronousQueue< E >::drainTo().

6.668.3.14 `template<typename E > virtual bool
decaf::util::concurrent::SynchronousQueue< E >::poll (
E & result) [inline, virtual]`

Retrieves and removes the head of this queue, if another thread is currently making an element available.

Parameters

result a reference to the value where the head of the **Queue** (p. 2507) should be copied to.

Returns

true if the head of the **Queue** (p. 2507) was copied to the result param or false if no value could be returned.

```

6.668.3.15  template<typename E > virtual void
             decaf::util::concurrent::SynchronousQueue< E >::put ( const E
             & value ) throw ( decaf::lang::exceptions::InterruptedException,
             decaf::lang::exceptions::NullPointerException, de-
             ccaf::lang::exceptions::IllegalArgumentException ) [inline,
             virtual]

```

Adds the specified element to this queue, waiting if necessary for another thread to receive it.

Parameters

value the element to add to the **Queue** (p.2507).

Exceptions

InterruptedException

NullPointerException

IllegalArgumentException

```

6.668.3.16  template<typename E > virtual int
             decaf::util::concurrent::SynchronousQueue< E
             >::remainingCapacity ( ) const [inline, virtual]

```

```

6.668.3.17  template<typename E > virtual bool
             decaf::util::concurrent::SynchronousQueue< E >::remove
             ( const E &value DECAF_UNUSED ) throw (
             lang::exceptions::UnsupportedOperationException,
             lang::exceptions::IllegalArgumentException ) [inline, virtual]

```

```

6.668.3.18  template<typename E > virtual bool
             decaf::util::concurrent::SynchronousQueue< E
             >::removeAll ( const Collection< E > &collection DECAF_UNUSED
             ) throw ( lang::exceptions::UnsupportedOperationException,
             lang::exceptions::IllegalArgumentException ) [inline, virtual]

```

```

6.668.3.19  template<typename E > virtual bool
             decaf::util::concurrent::SynchronousQueue< E
             >::retainAll ( const Collection< E > &collection DECAF_UNUSED
             ) throw ( lang::exceptions::UnsupportedOperationException,
             lang::exceptions::IllegalArgumentException ) [inline, virtual]

```

```

6.668.3.20  template<typename E > virtual std::size_t
             decaf::util::concurrent::SynchronousQueue< E >::size ( ) const
             [inline, virtual]

```

```

6.668.3.21  template<typename E > virtual E
             decaf::util::concurrent::SynchronousQueue< E >::take (
             ) throw ( decaf::lang::exceptions::InterruptedException ) [inline,
             virtual]

```

Retrieves and removes the head of this queue, waiting if necessary for another thread to insert it.

Returns

the head of this queue

Exceptions

InterruptedException

```
6.668.3.22  template<typename E > virtual std::vector<E>
             decaf::util::concurrent::SynchronousQueue< E >::toArray (    ) const
             [inline, virtual]
```

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**SynchronousQueue.h**

6.669 decaf::lang::System Class Reference

```
#include <src/main/decaf/lang/System.h>
```

Public Member Functions

- **System** ()
- virtual **~System** ()

Static Public Member Functions

- static const **util::Map**< std::string, std::string > & **getenv** () throw (lang::Exception)
Enumerates the system environment and returns a map of env variable names to the string values they hold.
- static std::string **getenv** (const std::string &name) throw (lang::Exception)
Reads an environment value from the system and returns it as a string object.
- static void **unsetenv** (const std::string &name) throw (lang::Exception)
Clears a set env value if one is set.
- static void **setenv** (const std::string &name, const std::string &value) throw (lang::Exception)
Sets the specified system property to the value given.
- static long long **currentTimeMillis** ()
Returns the current time in milliseconds.
- static long long **nanoTime** ()
Returns the current value of the most precise available system timer, in nanoseconds.
- static int **availableProcessors** ()
Returns the number of processors available for execution of Decaf Threads.

6.669.1 Constructor & Destructor Documentation

6.669.1.1 `decaf::lang::System::System ()`

6.669.1.2 `virtual decaf::lang::System::~~System ()` [inline, virtual]

6.669.2 Member Function Documentation

6.669.2.1 `static int decaf::lang::System::availableProcessors ()` [static]

Returns the number of processors available for execution of Decaf Threads.

This value may change during a particular execution of a Decaf based application. Applications that are sensitive to the number of available processors should therefore occasionally poll this property and adjust their resource usage appropriately.

Returns

the number of available processors.

6.669.2.2 `static long long decaf::lang::System::currentTimeMillis ()` [static]

Returns the current time in milliseconds.

Note that while the unit of time of the return value is a millisecond, the granularity of the value depends on the underlying operating system and may be larger. For example, many operating systems measure time in units of tens of milliseconds.

See the description of the class Date for a discussion of slight discrepancies that may arise between "computer time" and coordinated universal time (UTC).

Returns

the difference, measured in milliseconds, between the current time and midnight, January 1, 1970 UTC.

6.669.2.3 `static const util::Map<std::string, std::string>& decaf::lang::System::getenv ()` throw (`lang::Exception`) [static]

Enumerates the system environment and returns a map of env variable names to the string values they hold.

Returns

A Map of all environment variables.

Exceptions

Exception (p. 1476) if an error occurs

6.669.2.4 `static std::string decaf::lang::System::getenv (const std::string & name) throw (lang::Exception) [static]`

Reads an environment value from the system and returns it as a string object.

Parameters

name - the env var to read

Returns

a string with the value from the var or ""

Exceptions

an Exception (p. 1476) if an error occurs while reading the Env.

6.669.2.5 `static long long decaf::lang::System::nanoTime () [static]`

Returns the current value of the most precise available system timer, in nanoseconds.

This method can only be used to measure elapsed time and is not related to any other notion of system or wall-clock time. The value returned represents nanoseconds since some fixed but arbitrary time (perhaps in the future, so values may be negative). This method provides nanosecond precision, but not necessarily nanosecond accuracy. No guarantees are made about how frequently values change. Differences in successive calls that span greater than approximately 292 years (263 nanoseconds) will not accurately compute elapsed time due to numerical overflow.

For example, to measure how long some code takes to execute:

```
long long startTime = System::nanoTime() (p. 2955); // ... the code being measured ... long
long estimatedTime = System::nanoTime() (p. 2955) - startTime;
```

Returns

The current value of the system timer, in nanoseconds.

6.669.2.6 `static void decaf::lang::System::setenv (const std::string & name, const std::string & value) throw (lang::Exception) [static]`

Sets the specified system property to the value given.

Parameters

name - name of the env val to set

value - value to assign to name

Exceptions

an Exception (p. 1476) if an error occurs

6.669.2.7 `static void decaf::lang::System::unsetenv (const std::string & name)
throw (lang::Exception) [static]`

Clears a set env value if one is set.

Parameters

name - the env var to clear

Exceptions

an Exception (p. 1476) if an error occurs while reading the Env.

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/System.h`

6.670 activemq::threads::Task Class Reference

Represents a unit of work that requires one or more iterations to complete.

`#include <src/main/activemq/threads/Task.h>`

Inheritance diagram for `activemq::threads::Task`:

Public Member Functions

- `virtual ~Task ()`
- `virtual bool iterate ()=0`

Perform one iteration of work, returns true if the task needs to run again to complete or false to indicate that the task is now complete.

6.670.1 Detailed Description

Represents a unit of work that requires one or more iterations to complete.

Since

3.0

6.670.2 Constructor & Destructor Documentation

6.670.2.1 `virtual activemq::threads::Task::~~Task () [inline, virtual]`

6.670.3 Member Function Documentation

6.670.3.1 `virtual bool activemq::threads::Task::iterate () [pure virtual]`

Perform one iteration of work, returns true if the task needs to run again to complete or false to indicate that the task is now complete.

Returns

true if the task should be run again or false if the task has completed and the runner should wait for a wakeup call.

Implemented in `activemq::core::ActiveMQSessionExecutor` (p. 421), `activemq::transport::failover::BackupTransportPool` (p. 596), `activemq::transport::failover::CloseTransportsTask` (p. 953), and `activemq::transport::failover::FailoverTransport` (p. 1518).

The documentation for this class was generated from the following file:

- `src/main/activemq/threads/Task.h`

6.671 decaf::util::concurrent::TaskListener Class Reference

```
#include <src/main/decaf/util/concurrent/TaskListener.h>
```

Public Member Functions

- virtual `~TaskListener()`
- virtual void `onTaskComplete (lang::Runnable *task)=0`
Called when a queued task has completed, the task that finished is passed along for user consumption.
- virtual void `onTaskException (lang::Runnable *task, lang::Exception &ex)=0`
Called when a queued task has thrown an exception while being run.

6.671.1 Constructor & Destructor Documentation

6.671.1.1 virtual `decaf::util::concurrent::TaskListener::~~TaskListener()`
[inline, virtual]

6.671.2 Member Function Documentation

6.671.2.1 virtual void `decaf::util::concurrent::TaskListener::onTaskComplete (lang::Runnable * task)` [pure virtual]

Called when a queued task has completed, the task that finished is passed along for user consumption.

Parameters

task Runnable Pointer to the task that finished

6.671.2.2 virtual void `decaf::util::concurrent::TaskListener::onTaskException (lang::Runnable * task, lang::Exception & ex)` [pure virtual]

Called when a queued task has thrown an exception while being run.

The Callee should assume that this was an unrecoverable exception and that this task is now defunct.

Parameters

- task* Runnable Pointer to the task
- ex* The ActiveMQException that was thrown.

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**TaskListener.h**

6.672 activemq::threads::TaskRunner Class Reference

```
#include <src/main/activemq/threads/TaskRunner.h>
```

Inheritance diagram for activemq::threads::TaskRunner:

Public Member Functions

- virtual **~TaskRunner** ()
- virtual void **shutdown** (unsigned int timeout)=0
Shutdown after a timeout, does not guarantee that the task's iterate method has completed and the thread halted.
- virtual void **shutdown** ()=0
Shutdown once the task has finished and the TaskRunner's thread has exited.
- virtual void **wakeup** ()=0
*Signal the **TaskRunner** (p. 2958) to wakeup and execute another iteration cycle on the task, the **Task** (p. 2956) instance will be run until its iterate method has returned false indicating it is done.*

6.672.1 Constructor & Destructor Documentation

6.672.1.1 virtual **activemq::threads::TaskRunner::~TaskRunner** () [inline, virtual]

6.672.2 Member Function Documentation

6.672.2.1 virtual void **activemq::threads::TaskRunner::shutdown** (unsigned int *timeout*) [pure virtual]

Shutdown after a timeout, does not guarantee that the task's iterate method has completed and the thread halted.

Parameters

- timeout* - Time in Milliseconds to wait for the task to stop.

6.672.2.2 virtual void activemq::threads::TaskRunner::shutdown () [pure virtual]

Shutdown once the task has finished and the TaskRunner's thread has exited.

6.672.2.3 virtual void activemq::threads::TaskRunner::wakeup () [pure virtual]

Signal the **TaskRunner** (p. 2958) to wakeup and execute another iteration cycle on the task, the **Task** (p. 2956) instance will be run until its iterate method has returned false indicating it is done.

The documentation for this class was generated from the following file:

- src/main/activemq/threads/**TaskRunner.h**

6.673 decaf::net::TcpSocket Class Reference

Platform-independent implementation of the socket interface.

```
#include <src/main/decaf/net/TcpSocket.h>
```

Inheritance diagram for decaf::net::TcpSocket:

Public Member Functions

- **TcpSocket** () throw (SocketException)
Construct a non-connected socket.
- **TcpSocket** (SocketHandle socketHandle)
Construct a connected or bound socket based on given socket handle.
- virtual ~**TcpSocket** ()
Releases the socket handle but not gracefully shut down the connection.
- **SocketHandle** getSocketHandle ()
Gets the handle for the socket.
- void **connect** (const char *host, int port, int timeout) throw (SocketException)
Connects to the specified destination.
- virtual void **connect** (const char *host, int port) throw (SocketException)
Connects to the specified destination.
- virtual bool **isConnected** () const
Indicates whether or not this socket is connected to a destination.
- virtual **io::InputStream** * getInputStream ()
Gets the InputStream for this socket.

- virtual **io::OutputStream * getOutputStream ()**
Gets the OutputStream for this socket.
- virtual **int getSoLinger () const throw (SocketException)**
Gets the linger time.
- virtual **void setSoLinger (int linger) throw (SocketException)**
Sets the linger time.
- virtual **bool getKeepAlive () const throw (SocketException)**
Gets the keep alive flag.
- virtual **void setKeepAlive (bool keepAlive) throw (SocketException)**
Enables/disables the keep alive flag.
- virtual **int getReceiveBufferSize () const throw (SocketException)**
Gets the receive buffer size.
- virtual **void setReceiveBufferSize (int size) throw (SocketException)**
Sets the receive buffer size.
- virtual **bool getReuseAddress () const throw (SocketException)**
Gets the reuse address flag.
- virtual **void setReuseAddress (bool reuse) throw (SocketException)**
Sets the reuse address flag.
- virtual **int getSendBufferSize () const throw (SocketException)**
Gets the send buffer size.
- virtual **void setSendBufferSize (int size) throw (SocketException)**
Sets the send buffer size.
- virtual **int getSoTimeout () const throw (SocketException)**
Gets the timeout for socket operations.
- virtual **void setSoTimeout (int timeout) throw (SocketException)**
Sets the timeout for socket operations.
- virtual **void close () throw (decaf::io::IOException)**
Closes this object and deallocates the appropriate resources.
- virtual **bool getTcpNoDelay () const throw (lang::Exception)**
Gets the Status of the TCP_NODELAY param for this socket as a Bool.
- virtual **void setTcpNoDelay (bool value) throw (lang::Exception)**
Sets the Status of the TCP_NODELAY param for this socket as a Bool.

Protected Member Functions

- void **checkResult** (apr_status_t value) const throw (SocketException)

6.673.1 Detailed Description

Platform-independent implementation of the socket interface.

6.673.2 Constructor & Destructor Documentation

6.673.2.1 decaf::net::TcpSocket::TcpSocket () throw (SocketException)

Construct a non-connected socket.

Exceptions

SocketException (p. 2793) thrown one windows if the static initialization call to WSAS-tartup was not successful.

6.673.2.2 decaf::net::TcpSocket::TcpSocket (SocketHandle socketHandle)

Construct a connected or bound socket based on given socket handle.

Parameters

socketHandle a socket handle to wrap in the object

6.673.2.3 virtual decaf::net::TcpSocket::~~TcpSocket () [virtual]

Releases the socket handle but not gracefully shut down the connection.

6.673.3 Member Function Documentation

6.673.3.1 void decaf::net::TcpSocket::checkResult (apr_status_t value) const throw (SocketException) [protected]

6.673.3.2 virtual void decaf::net::TcpSocket::close () throw (decaf::io::IOException) [virtual]

Closes this object and deallocates the appropriate resources.

Exceptions

IOException

Implements **decaf::io::Closeable** (p. 951).

6.673.3.3 `void decaf::net::TcpSocket::connect (const char * host, int port, int timeout) throw (SocketException)`

Connects to the specified destination.

Closes this socket if connected to another destination.

Parameters

host The host of the server to connect to.

port The port of the server to connect to.

timeout of socket in microseconds

Exceptions

SocketException (p. 2793) Thrown if a failure occurred in the connect.

6.673.3.4 `virtual void decaf::net::TcpSocket::connect (const char * host, int port) throw (SocketException) [inline, virtual]`

Connects to the specified destination.

Closes this socket if connected to another destination.

Parameters

host The host of the server to connect to.

port The port of the server to connect to.

Exceptions

SocketException (p. 2793) Thrown if a failure occurred in the connect.

Implements `decaf::net::Socket` (p. 2787).

6.673.3.5 `virtual io::InputStream* decaf::net::TcpSocket::getInputStream () [virtual]`

Gets the InputStream for this socket.

Returns

The InputStream for this socket. NULL if not connected.

Implements `decaf::net::Socket` (p. 2788).

6.673.3.6 `virtual bool decaf::net::TcpSocket::getKeepAlive () const throw (SocketException) [virtual]`

Gets the keep alive flag.

Returns

True if keep alive is enabled.

Exceptions

SocketException (p. 2793) if the operation fails.

Implements **decaf::net::Socket** (p. 2788).

6.673.3.7 `virtual io::OutputStream* decaf::net::TcpSocket::getOutputStream ()`
[virtual]

Gets the OutputStream for this socket.

Returns

the OutputStream for this socket. NULL if not connected.

Implements **decaf::net::Socket** (p. 2788).

6.673.3.8 `virtual int decaf::net::TcpSocket::getReceiveBufferSize () const throw (SocketException)` [virtual]

Gets the receive buffer size.

Returns

the receive buffer size in bytes.

Exceptions

SocketException (p. 2793) if the operation fails.

Implements **decaf::net::Socket** (p. 2788).

6.673.3.9 `virtual bool decaf::net::TcpSocket::getReuseAddress () const throw (SocketException)` [virtual]

Gets the reuse address flag.

Returns

True if the address can be reused.

Exceptions

SocketException (p. 2793) if the operation fails.

Implements **decaf::net::Socket** (p. 2789).

6.673.3.10 `virtual int decaf::net::TcpSocket::getSendBufferSize () const throw (SocketException)` [virtual]

Gets the send buffer size.

Returns

the size in bytes of the send buffer.

Exceptions

SocketException (p. 2793) if the operation fails.

Implements **decaf::net::Socket** (p. 2789).

6.673.3.11 SocketHandle decaf::net::TcpSocket::getSocketHandle () [inline]

Gets the handle for the socket.

Returns

SocketHabler for this **Socket** (p. 2786), can be NULL

6.673.3.12 virtual int decaf::net::TcpSocket::getSoLinger () const throw (SocketException) [virtual]

Gets the linger time.

Returns

The linger time in seconds.

Exceptions

SocketException (p. 2793) if the operation fails.

Implements **decaf::net::Socket** (p. 2789).

6.673.3.13 virtual int decaf::net::TcpSocket::getSoTimeout () const throw (SocketException) [virtual]

Gets the timeout for socket operations.

Returns

The timeout in milliseconds for socket operations.

Exceptions

SocketException (p. 2793) Thrown if unable to retrieve the information.

Implements **decaf::net::Socket** (p. 2790).

6.673.3.14 virtual bool decaf::net::TcpSocket::getTcpNoDelay () const throw (lang::Exception) [virtual]

Gets the Status of the TCP_NODELAY param for this socket as a Bool.

Returns

true if TCP_NODELAY is enabled

Exceptions

Exception

6.673.3.15 `virtual bool decaf::net::TcpSocket::isConnected () const [inline, virtual]`

Indicates whether or not this socket is connected to a destination.

Returns

true if connected

Implements **decaf::net::Socket** (p. 2790).

6.673.3.16 `virtual void decaf::net::TcpSocket::setKeepAlive (bool keepAlive) throw (SocketException) [virtual]`

Enables/disables the keep alive flag.

Parameters

keepAlive If true, enables the flag.

Exceptions

SocketException (p. 2793) if the operation fails.

Implements **decaf::net::Socket** (p. 2790).

6.673.3.17 `virtual void decaf::net::TcpSocket::setReceiveBufferSize (int size) throw (SocketException) [virtual]`

Sets the receive buffer size.

Parameters

size Number of bytes to set the receive buffer to.

Exceptions

SocketException (p. 2793) if the operation fails.

Implements **decaf::net::Socket** (p. 2790).

6.673.3.18 `virtual void decaf::net::TcpSocket::setReuseAddress (bool reuse) throw (SocketException) [virtual]`

Sets the reuse address flag.

Parameters

reuse If true, sets the flag.

Exceptions

SocketException (p. 2793) if the operation fails.

Implements **decaf::net::Socket** (p. 2791).

6.673.3.19 `virtual void decaf::net::TcpSocket::setSendBufferSize (int size) throw (SocketException)` [virtual]

Sets the send buffer size.

Parameters

size The number of bytes to set the send buffer to.

Exceptions

SocketException (p. 2793) if the operation fails.

Implements `decaf::net::Socket` (p. 2791).

6.673.3.20 `virtual void decaf::net::TcpSocket::setSoLinger (int linger) throw (SocketException)` [virtual]

Sets the linger time.

Parameters

linger The linger time in seconds. If 0, linger is off.

Exceptions

SocketException (p. 2793) if the operation fails.

Implements `decaf::net::Socket` (p. 2791).

6.673.3.21 `virtual void decaf::net::TcpSocket::setSoTimeout (int timeout) throw (SocketException)` [virtual]

Sets the timeout for socket operations.

Parameters

timeout The timeout in milliseconds for socket operations.

Exceptions

SocketException (p. 2793) Thrown if unable to set the information.

Implements `decaf::net::Socket` (p. 2792).

6.673.3.22 `virtual void decaf::net::TcpSocket::setTcpNoDelay (bool value) throw (lang::Exception)` [virtual]

Sets the Status of the TCP_NODELAY param for this socket as a Bool.

Parameters

value - true if TCP_NODELAY is to be enabled

Exceptions

Exception

The documentation for this class was generated from the following file:

- `src/main/decaf/net/TcpSocket.h`

6.674 activemq::transport::tcp::TcpTransport Class Reference

Implements a TCP/IP based transport filter, this transport is meant to wrap an instance of an **IOTransport** (p. 1709).

```
#include <src/main/activemq/transport/tcp/TcpTransport.h>
```

Inheritance diagram for `activemq::transport::tcp::TcpTransport`:

Public Member Functions

- **TcpTransport** (const **decaf::util::Properties** &properties, const **Pointer**< **Transport** > &next)
Constructor.
- **TcpTransport** (const **decaf::net::URI** &uri, const **decaf::util::Properties** &properties, const **Pointer**< **Transport** > &next)
Constructor.
- virtual **~TcpTransport** ()
- virtual void **close** () throw (**decaf::io::IOException**)
Delegates to the superclass and then closes the socket.
- virtual bool **isFaultTolerant** () const
*Is this **Transport** (p. 3066) fault tolerant, meaning that it will reconnect to a broker on disconnect.*
- virtual bool **isConnected** () const
*Is the **Transport** (p. 3066) Connected to its Broker.*
- virtual bool **isClosed** () const
*Has the **Transport** (p. 3066) been shutdown and no longer usable.*

6.674.1 Detailed Description

Implements a TCP/IP based transport filter, this transport is meant to wrap an instance of an **IOTransport** (p. 1709). The lower level transport should take care of managing stream reads and writes.

6.674.2 Constructor & Destructor Documentation

6.674.2.1 `activemq::transport::tcp::TcpTransport::TcpTransport (const decaf::util::Properties & properties, const Pointer< Transport > & next)`

Constructor.

Parameters

properties the configuration properties for this transport

next the next transport in the chain

6.674.2.2 `activemq::transport::tcp::TcpTransport::TcpTransport (const decaf::net::URI & uri, const decaf::util::Properties & properties, const Pointer< Transport > & next)`

Constructor.

Parameters

uri - The URI containing the host to connect to.

properties the configuration properties for this transport

next the next transport in the chain

6.674.2.3 `virtual activemq::transport::tcp::TcpTransport::~~TcpTransport ()`
[virtual]

6.674.3 Member Function Documentation

6.674.3.1 `virtual void activemq::transport::tcp::TcpTransport::close () throw (decaf::io::IOException)` [virtual]

Delegates to the superclass and then closes the socket.

Exceptions

IOException if errors occur.

Reimplemented from `activemq::transport::TransportFilter` (p. 3075).

6.674.3.2 `virtual bool activemq::transport::tcp::TcpTransport::isClosed () const`
[inline, virtual]

Has the **Transport** (p. 3066) been shutdown and no longer usable.

Returns

true if the **Transport** (p. 3066)

Reimplemented from `activemq::transport::TransportFilter` (p. 3076).

6.674.3.3 virtual bool activemq::transport::tcp::TcpTransport::isConnected () const [inline, virtual]

Is the **Transport** (p. 3066) Connected to its Broker.

Returns

true if a connection has been made.

Reimplemented from **activemq::transport::TransportFilter** (p. 3076).

6.674.3.4 virtual bool activemq::transport::tcp::TcpTransport::isFaultTolerant () const [inline, virtual]

Is this **Transport** (p. 3066) fault tolerant, meaning that it will reconnect to a broker on disconnect.

Returns

true if the **Transport** (p. 3066) is fault tolerant.

Reimplemented from **activemq::transport::TransportFilter** (p. 3077).

The documentation for this class was generated from the following file:

- src/main/activemq/transport/tcp/**TcpTransport.h**

6.675 activemq::transport::tcp::TcpTransportFactory Class Reference

Factory Responsible for creating the **TcpTransport** (p. 2967).

```
#include <src/main/activemq/transport/tcp/TcpTransportFactory.h>
```

Inheritance diagram for **activemq::transport::tcp::TcpTransportFactory**:

Public Member Functions

- virtual **~TcpTransportFactory** ()
- virtual **Pointer< Transport > create** (const **decaf::net::URI** &location) throw (exceptions::ActiveMQException)
*Creates a fully configured **Transport** (p. 3066) instance which could be a chain of filters and transports.*
- virtual **Pointer< Transport > createComposite** (const **decaf::net::URI** &location) throw (exceptions::ActiveMQException)
*Creates a slimed down **Transport** (p. 3066) instance which can be used in composite transport instances.*

Protected Member Functions

- virtual **Pointer< Transport > doCreateComposite** (const **decaf::net::URI** &location, const **Pointer< wireformat::WireFormat >** &wireFormat, const **decaf::util::Properties** &properties) throw (exceptions::ActiveMQException)

*Creates a slimed down **Transport** (p. 3066) instance which can be used in composite transport instances.*

6.675.1 Detailed Description

Factory Responsible for creating the **TcpTransport** (p. 2967).

6.675.2 Constructor & Destructor Documentation

- 6.675.2.1** virtual
activemq::transport::tcp::TcpTransportFactory::~~TcpTransportFactory (
) [inline, virtual]

6.675.3 Member Function Documentation

- 6.675.3.1** virtual **Pointer<Transport> activemq::transport::tcp::TcpTransportFactory::create**
 (const **decaf::net::URI** & *location*) throw (exceptions::ActiveMQException) [virtual]

Creates a fully configured **Transport** (p. 3066) instance which could be a chain of filters and transports.

Parameters

location - URI location to connect to plus any properties to assign.

Exceptions

ActiveMQException if an error occurs

Implements **activemq::transport::TransportFactory** (p. 3072).

- 6.675.3.2** virtual **Pointer<Transport> activemq::transport::tcp::TcpTransportFactory::createComposite** (const **decaf::net::URI** & *location*) throw (exceptions::ActiveMQException) [virtual]

Creates a slimed down **Transport** (p. 3066) instance which can be used in composite transport instances.

Parameters

location - URI location to connect to plus any properties to assign.

Exceptions

ActiveMQException if an error occurs

Implements **activemq::transport::TransportFactory** (p. 3072).

6.675.3.3 `virtual Pointer<Transport> activemq::transport::tcp::TcpTransportFactory::doCreateComposite (const decaf::net::URI & location, const Pointer< wireformat::WireFormat > & wireFormat, const decaf::util::Properties & properties) throw (exceptions::ActiveMQException)` [protected, virtual]

Creates a slimed down **Transport** (p. 3066) instance which can be used in composite transport instances.

Parameters

location - URI location to connect to.

wireFormat - the assigned WireFormat for the new **Transport** (p. 3066).

properties - Properties to apply to the transport.

Returns

new Pointer to a **TcpTransport** (p. 2967).

Exceptions

ActiveMQException if an error occurs

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/tcp/TcpTransportFactory.h`

6.676 cms::TemporaryQueue Class Reference

Defines a Temporary **Queue** (p. 2510) based **Destination** (p. 1387).

```
#include <src/main/cms/TemporaryQueue.h>
```

Inheritance diagram for cms::TemporaryQueue:

Public Member Functions

- `virtual ~TemporaryQueue ()`
- `virtual std::string getQueueName () const =0 throw (CMSEException)`
Gets the name of this queue.
- `virtual void destroy ()=0 throw (CMSEException)`
*Destroy's the Temporary **Destination** (p. 1387) at the Provider.*

6.676.1 Detailed Description

Defines a Temporary **Queue** (p.2510) based **Destination** (p.1387). A **TemporaryQueue** (p.2971) is a special type of **Queue** (p.2510) **Destination** (p.1387) that can only be consumed from the **Connection** (p.1052) which created it. TemporaryQueues are most commonly used as the reply to address for Message's that implement the request response pattern.

A **TemporaryQueue** (p.2971) is guaranteed to exist at the Provider only for the lifetime of the **Connection** (p.1052) that created it.

Since

1.0

6.676.2 Constructor & Destructor Documentation

6.676.2.1 `virtual cms::TemporaryQueue::~TemporaryQueue () [inline, virtual]`

6.676.3 Member Function Documentation

6.676.3.1 `virtual void cms::TemporaryQueue::destroy () throw (CMSEException) [pure virtual]`

Destroy's the Temporary **Destination** (p.1387) at the Provider.

Exceptions

CMSEException (p. 960) - if an internal error occurs.

Implemented in `activemq::commands::ActiveMQTempQueue` (p.479).

6.676.3.2 `virtual std::string cms::TemporaryQueue::getQueueName () const throw (CMSEException) [pure virtual]`

Gets the name of this queue.

Returns

The queue name.

Exceptions

CMSEException (p. 960) - if an internal error occurs.

Implemented in `activemq::commands::ActiveMQTempQueue` (p.480).

The documentation for this class was generated from the following file:

- `src/main/cms/TemporaryQueue.h`

6.677 cms::TemporaryTopic Class Reference

Defines a Temporary **Topic** (p. 3014) based **Destination** (p. 1387).

```
#include <src/main/cms/TemporaryTopic.h>
```

Inheritance diagram for cms::TemporaryTopic:

Public Member Functions

- virtual **~TemporaryTopic** ()
- virtual std::string **getTopicName** () const =0 throw (CMSEException)
Gets the name of this topic.
- virtual void **destroy** ()=0 throw (CMSEException)
*Destroy's the Temporary **Destination** (p. 1387) at the Provider.*

6.677.1 Detailed Description

Defines a Temporary **Topic** (p. 3014) based **Destination** (p. 1387). A **TemporaryTopic** (p. 2973) is a special type of **Topic** (p. 3014) **Destination** (p. 1387) that can only be consumed from the **Connection** (p. 1052) which created it. TemporaryTopics are most commonly used as the reply to address for Message's that implement the request response pattern.

A **TemporaryTopic** (p. 2973) is guaranteed to exist at the Provider only for the lifetime of the **Connection** (p. 1052) that created it.

Since

1.0

6.677.2 Constructor & Destructor Documentation

6.677.2.1 virtual cms::TemporaryTopic::~~TemporaryTopic () [inline, virtual]

6.677.3 Member Function Documentation

6.677.3.1 virtual void cms::TemporaryTopic::destroy () throw (CMSEException) [pure virtual]

Destroy's the Temporary **Destination** (p. 1387) at the Provider.

Exceptions

CMSEException (p. 960)

Implemented in **activemq::commands::ActiveMQTempTopic** (p. 503).

6.677.3.2 `virtual std::string cms::TemporaryTopic::getTopicName () const throw (CMSEException)` [pure virtual]

Gets the name of this topic.

Returns

The topic name.

Exceptions

CMSEException (p. 960) - if an internal error occurs.

Implemented in `activemq::commands::ActiveMQTempTopic` (p. 504).

The documentation for this class was generated from the following file:

- `src/main/cms/TemporaryTopic.h`

6.678 cms::TextMessage Class Reference

Interface for a text message.

```
#include <src/main/cms/TextMessage.h>
```

Inheritance diagram for `cms::TextMessage`:

Public Member Functions

- `virtual ~TextMessage ()`
- `virtual std::string getText () const =0 throw (cms::CMSEException)`
Gets the message character buffer.
- `virtual void setText (const char *msg)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)`
Sets the message contents, does not take ownership of the passed char, but copies it instead.*
- `virtual void setText (const std::string &msg)=0 throw (cms::MessageNotWriteableException, cms::CMSEException)`
Sets the message contents.

6.678.1 Detailed Description

Interface for a text message. A `TextMessage` (p. 2974) can contain any Text based pay load such as an XML Document or other Text based document.

Like all Messages, a `TextMessage` (p. 2974) is received in Read-Only mode, any attempt to write to the `Message` (p. 2036) will result in a `MessageNotWritableException` being thrown until the `clearBody` method is called which will erase the contents and place the message back in a read / write mode.

Since

1.0

6.678.2 Constructor & Destructor Documentation

6.678.2.1 virtual cms::TextMessage::~TextMessage () [inline, virtual]

6.678.3 Member Function Documentation

6.678.3.1 virtual std::string cms::TextMessage::getText () const throw (cms::CMSEException) [pure virtual]

Gets the message character buffer.

Returns

The message character buffer.

Exceptions

CMSEException (p. 960) - if an internal error occurs.

6.678.3.2 virtual void cms::TextMessage::setText (const std::string & msg) throw (cms::MessageNotWriteableException, cms::CMSEException) [pure virtual]

Sets the message contents.

Parameters

msg The message buffer.

Exceptions

CMSEException (p. 960) - if an internal error occurs.

MessageNotWriteableException (p. 2188) - if the message is in read-only mode..

6.678.3.3 virtual void cms::TextMessage::setText (const char * msg) throw (cms::MessageNotWriteableException, cms::CMSEException) [pure virtual]

Sets the message contents, does not take ownership of the passed char*, but copies it instead.

Parameters

msg The message buffer.

Exceptions

CMSEException (p. 960) - if an internal error occurs.

MessageNotWriteableException (p. 2188) - if the message is in read-only mode..

The documentation for this class was generated from the following file:

- `src/main/cms/TextMessage.h`

6.679 decaf::util::concurrent::ThreadFactory Class Reference

public interface **ThreadFactory** (p.2976)

```
#include <src/main/decaf/util/concurrent/ThreadFactory.h>
```

Public Member Functions

- virtual `~ThreadFactory()`
- virtual `decaf::lang::Thread * newThread (decaf::lang::Runnable *r)=0`
Constructs a new Thread.

6.679.1 Detailed Description

public interface **ThreadFactory** (p.2976) An object that creates new threads on demand. Using thread factories removes hardwiring of calls to new Thread, enabling applications to use special thread subclasses, priorities, etc.

The simplest implementation of this interface is just:

```
class SimpleThreadFactory : public ThreadFactory (p.2976) { public: Thread* newThread(
Runnable* r ) { return new Thread(r); } }
```

The `Executors.defaultThreadFactory()` method provides a more useful simple implementation, that sets the created thread context to known values before returning it.

Since

1.0

6.679.2 Constructor & Destructor Documentation

6.679.2.1 virtual `decaf::util::concurrent::ThreadFactory::~~ThreadFactory ()`
`[inline, virtual]`

6.679.3 Member Function Documentation

6.679.3.1 virtual `decaf::lang::Thread* decaf::util::concurrent::ThreadFactory::newThread (decaf::lang::Runnable * r)` `[pure virtual]`

Constructs a new Thread.

Implementations may also initialize priority, name, daemon status, ThreadGroup, etc. The pointer passed is still owned by the caller and is not deleted by the Thread object. The caller owns the returned Thread object and must delete it when finished.

Parameters

- r* A pointer to a Runnable instance to be executed by new Thread instance returned.

Returns

constructed thread, or NULL if the request to create a thread is rejected the caller owns the returned pointer.

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**ThreadFactory.h**

6.680 decaf::lang::ThreadGroup Class Reference

```
#include <src/main/decaf/lang/ThreadGroup.h>
```

Public Member Functions

- ThreadGroup** ()
- virtual **~ThreadGroup** ()

6.680.1 Detailed Description

Since

1.0

6.680.2 Constructor & Destructor Documentation

6.680.2.1 decaf::lang::ThreadGroup::ThreadGroup ()

6.680.2.2 virtual decaf::lang::ThreadGroup::~~ThreadGroup () [virtual]

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**ThreadGroup.h**

6.681 decaf::util::concurrent::ThreadPool Class Reference

Defines a Thread Pool object that implements the functionality of pooling threads to perform user tasks.

```
#include <src/main/decaf/util/concurrent/ThreadPool.h>
```

Inheritance diagram for decaf::util::concurrent::ThreadPool:

Public Types

- typedef std::pair< **lang::Runnable** *, **TaskListener** * > **Task**

Public Member Functions

- **ThreadPool ()**
- virtual **~ThreadPool ()**
- virtual void **queueTask (Task task)** throw (lang::Exception)
Queue (p. 2507) a task to be completed by one of the Pooled Threads.
- virtual **Task deQueueTask ()** throw (lang::Exception)
DeQueue a task to be completed by one of the Pooled Threads.
- virtual std::size_t **getPoolSize ()** const
Returns the current number of Threads in the Pool, this is how many there are now, not how many are active or the max number that might exist.
- virtual std::size_t **getBacklog ()** const
Returns the current backlog of items in the tasks queue, this is how much work is still waiting to get done.
- virtual void **reserve (std::size_t size)**
Ensures that there is at least the specified number of Threads allocated to the pool.
- virtual std::size_t **getMaxThreads ()** const
Get the Max Number of Threads this Pool can contain.
- virtual void **setMaxThreads (std::size_t maxThreads)**
Sets the Max number of threads this pool can contain.
- virtual std::size_t **getBlockSize ()** const
Gets the Max number of threads that can be allocated at a time when new threads are needed.
- virtual void **setBlockSize (std::size_t blockSize)**
Sets the Max number of Threads that can be allocated at a time when the Thread Pool determines that more Threads are needed.
- virtual std::size_t **getFreeThreadCount ()** const
Returns the current number of available threads in the pool, threads that are performing a user task are considered unavailable.
- virtual void **onTaskStarted (PooledThread *thread)**
Called by a pooled thread when it is about to begin executing a new task.
- virtual void **onTaskCompleted (PooledThread *thread)**
Called by a pooled thread when it has completed a task and is going back to waiting for another task to run, this will increment the free threads counter.
- virtual void **onTaskException (PooledThread *thread, lang::Exception &ex)**
*Called by a pooled thread when it has encountered an exception while running a user task, after receiving this notification the callee should assume that the **PooledThread** (p. 2364) is now no longer running.*

Static Public Member Functions

- static **ThreadPool** * **getInstance** ()
Return the one and only Thread Pool instance.

Static Public Attributes

- static const size_t **DEFAULT_MAX_POOL_SIZE** = 10
- static const size_t **DEFAULT_MAX_BLOCK_SIZE** = 3

6.681.1 Detailed Description

Defines a Thread Pool object that implements the functionality of pooling threads to perform user tasks. The Thread Pool has max size that it will grow to. The thread pool allocates threads in blocks. When there are no waiting worker threads and a task is queued then a new batch is allocated. The user can specify the size of the blocks, otherwise a default value is used.

When the user queues a task they must also queue a listener to be notified when the task has completed, this provides the user with a mechanism to know when a task object can be freed.

To have the Thread Pool perform a task, the user enqueue's an object that implements the **Runnable** interface and one of the worker threads will execute it in its thread context.

6.681.2 Member Typedef Documentation

- 6.681.2.1** `typedef std::pair<lang::Runnable*, TaskListener*>`
 `decaf::util::concurrent::ThreadPool::Task`

6.681.3 Constructor & Destructor Documentation

- 6.681.3.1** `decaf::util::concurrent::ThreadPool::ThreadPool ()`
- 6.681.3.2** `virtual decaf::util::concurrent::ThreadPool::~~ThreadPool ()` [virtual]

6.681.4 Member Function Documentation

- 6.681.4.1** `virtual Task decaf::util::concurrent::ThreadPool::deQueueTask ()`
 `throw (lang::Exception)` [virtual]

DeQueue a task to be completed by one of the Pooled Threads.

A caller of this method will block until there is something in the tasks queue, therefore care must be taken when calling this function. Normally clients of **ThreadPool** (p. 2977) don't use this, only the **PooledThread** (p. 2364) objects owned by this **ThreadPool** (p. 2977).

Returns

object that derives from **Runnable**

Exceptions

ActiveMQException

6.681.4.2 `virtual std::size_t decaf::util::concurrent::ThreadPool::getBacklog ()`
`const [inline, virtual]`

Returns the current backlog of items in the tasks queue, this is how much work is still waiting to get done.

Returns

number of outstanding tasks.

6.681.4.3 `virtual std::size_t decaf::util::concurrent::ThreadPool::getBlockSize ()`
`const [inline, virtual]`

Gets the Max number of threads that can be allocated at a time when new threads are needed.

Returns

max Thread Block Size

6.681.4.4 `virtual std::size_t decaf::util::concurrent::ThreadPool::getFreeThreadCount ()`
`const [inline, virtual]`

Returns the current number of available threads in the pool, threads that are performing a user task are considered unavailable.

This value could change immediately after calling as Threads could finish right after and be available again. This is informational only.

Returns

total free threads

6.681.4.5 `static ThreadPool* decaf::util::concurrent::ThreadPool::getInstance ()`
`[static]`

Return the one and only Thread Pool instance.

Returns

The Thread Pool Pointer

6.681.4.6 `virtual std::size_t decaf::util::concurrent::ThreadPool::getMaxThreads ()`
`const [inline, virtual]`

Get the Max Number of Threads this Pool can contain.

Returns

max size

6.681.4.7 `virtual std::size_t decaf::util::concurrent::ThreadPool::getPoolSize ()`
`const` [inline, virtual]

Returns the current number of Threads in the Pool, this is how many there are now, not how many are active or the max number that might exist.

Returns

integer number of threads in existence.

6.681.4.8 `virtual void decaf::util::concurrent::ThreadPool::onTaskCompleted (`
`PooledThread * thread)` [virtual]

Called by a pooled thread when it has completed a task and is going back to waiting for another task to run, this will increment the free threads counter.

Parameters

thread Pointer the the Pooled Thread that is making this call.

Implements `decaf::util::concurrent::PooledThreadListener` (p. 2367).

6.681.4.9 `virtual void decaf::util::concurrent::ThreadPool::onTaskException (`
`PooledThread * thread, lang::Exception & ex)` [virtual]

Called by a pooled thread when it has encountered an exception while running a user task, after receiving this notification the callee should assume that the **PooledThread** (p. 2364) is now no longer running.

Parameters

thread Pointer to the Pooled Thread that is making this call

ex The Exception that occurred.

Implements `decaf::util::concurrent::PooledThreadListener` (p. 2367).

6.681.4.10 `virtual void decaf::util::concurrent::ThreadPool::onTaskStarted (`
`PooledThread * thread)` [virtual]

Called by a pooled thread when it is about to begin executing a new task.

This will decrement the available threads counter so that this object knows when there are no more free threads and must create new ones.

Parameters

thread Pointer to the Pooled Thread that is making this call

Implements `decaf::util::concurrent::PooledThreadListener` (p. 2367).

6.681.4.11 `virtual void decaf::util::concurrent::ThreadPool::queueTask (Task task) throw (lang::Exception)` [virtual]

Queue (p. 2507) a task to be completed by one of the Pooled Threads.

tasks are serviced as soon as a `PooledThread` (p. 2364) is available to run it.

Parameters

task object that derives from Runnable

Exceptions

ActiveMQException

6.681.4.12 `virtual void decaf::util::concurrent::ThreadPool::reserve (std::size_t size)` [virtual]

Ensures that there is at least the specified number of Threads allocated to the pool.

If the size is greater than the MAX number of threads in the pool, then only MAX threads are reserved. If the size is smaller than the number of threads currently in the pool, than nothing is done.

Parameters

size the number of threads to reserve.

6.681.4.13 `virtual void decaf::util::concurrent::ThreadPool::setBlockSize (std::size_t blockSize)` [virtual]

Sets the Max number of Threads that can be allocated at a time when the Thread Pool determines that more Threads are needed.

Parameters

blockSize Max Thread Block Size

6.681.4.14 `virtual void decaf::util::concurrent::ThreadPool::setMaxThreads (std::size_t maxThreads)` [virtual]

Sets the Max number of threads this pool can contain.

if this value is smaller than the current size of the pool nothing is done.

Parameters

maxThreads total number of threads that can be pooled

6.681.5 Field Documentation

6.681.5.1 `const size_t decaf::util::concurrent::ThreadPool::DEFAULT_MAX_BLOCK_SIZE = 3` [static]

6.681.5.2 `const size_t decaf::util::concurrent::ThreadPool::DEFAULT_MAX_POOL_SIZE = 10` [static]

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/ThreadPool.h`

6.682 decaf::lang::Throwable Class Reference

This class represents an error that has occurred.

```
#include <src/main/decaf/lang/Throwable.h>
```

Inheritance diagram for decaf::lang::Throwable:

Public Member Functions

- **Throwable** () throw ()
- virtual **~Throwable** () throw ()
- virtual std::string **getMessage** () const =0
Gets the cause of the error, if no message was provided to the instance of this interface but a cause was then the value cause.getMessage is then returned.
- virtual const std::exception * **getCause** () const =0
Gets the exception that caused this one to be thrown, this allows for chaining of exceptions in the case of a method that throws only a particular exception but wishes to allow for the real causal exception to be passed only in case the caller knows about that type of exception and wishes to respond to it.
- virtual void **initCause** (const std::exception *cause)=0
Initializes the contained cause exception with the one given.
- virtual void **setMark** (const char *file, const int lineNumber)=0
Adds a file/line number to the stack trace.
- virtual **Throwable** * **clone** () const =0
Clones this exception.
- virtual std::vector< std::pair< std::string, int > > **getStackTrace** () const =0
Provides the stack trace for every point where this exception was caught, marked, and rethrown.
- virtual void **printStackTrace** () const =0
Prints the stack trace to std::err.

- virtual void **printStackTrace** (std::ostream &stream) const =0

Prints the stack trace to the given output stream.

- virtual std::string **getStackTraceString** () const =0

Gets the stack trace as one contiguous string.

6.682.1 Detailed Description

This class represents an error that has occurred. All Exceptions in the Decaf library should extend from this or from the **Exception** (p. 1476) class in order to ensure that all Decaf Exceptions are interchangeable with the std::exception class.

Throwable (p. 2983) can wrap another **Throwable** (p. 2983) as the cause if the error being thrown. The user can inspect the cause by calling **getCause**, the pointer returned is the property of the **Throwable** (p. 2983) instance and will be deleted when it is deleted or goes out of scope.

Since

1.0

6.682.2 Constructor & Destructor Documentation

6.682.2.1 **decaf::lang::Throwable::Throwable** () throw () [inline]

6.682.2.2 **virtual decaf::lang::Throwable::~~Throwable** () throw () [inline, virtual]

6.682.3 Member Function Documentation

6.682.3.1 **virtual Throwable* decaf::lang::Throwable::clone** () const [pure virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

Copy of this **Exception** (p. 1476) object

Implemented in **activemq::exceptions::ActiveMQException** (p. 271), **activemq::exceptions::BrokerException** (p. 691), **decaf::io::EOFException** (p. 1476), **decaf::io::InterruptedIOException** (p. 1695), **decaf::io::IOException** (p. 1709), **decaf::io::UnsupportedEncodingException** (p. 3092), **decaf::io::UTFDataFormatException** (p. 3141), **decaf::lang::Exception** (p. 1479), **decaf::lang::exceptions::ClassCastException** (p. 950), **decaf::lang::exceptions::IllegalArgumentException** (p. 1615), **decaf::lang::exceptions::IllegalMonitorStateException** (p. 1618), **decaf::lang::exceptions::IllegalStateException** (p. 1620), **decaf::lang::exceptions::IllegalThreadStateException** (p. 1624), **decaf::lang::exceptions::IndexOutOfBoundsException** (p. 1629), **de-**

cafe::lang::exceptions::InterruptedException (p. 1693), **de-**
cafe::lang::exceptions::InvalidStateException (p. 1706), **de-**
cafe::lang::exceptions::NoSuchElementException (p. 2276), **de-**
cafe::lang::exceptions::NullPointerException (p. 2281), **de-**
cafe::lang::exceptions::NumberFormatException (p. 2286),
decaf::lang::exceptions::RuntimeException (p. 2646), **de-**
cafe::lang::exceptions::UnsupportedOperationException (p. 3096), **de-**
cafe::net::BindException (p. 665), **decaf::net::ConnectException** (p. 1051), **de-**
cafe::net::HttpRetryException (p. 1613), **decaf::net::MalformedURLException** (p. 1970),
decaf::net::NoRouteToHostException (p. 2271), **decaf::net::PortUnreachableException**
 (p. 2370), **decaf::net::ProtocolException** (p. 2506), **decaf::net::SocketException** (p. 2794),
decaf::net::SocketTimeoutException (p. 2811), **decaf::net::UnknownHostException**
 (p. 3087), **decaf::net::UnknownServiceException** (p. 3089), **de-**
cafe::net::URISyntaxException (p. 3125), **decaf::nio::BufferOverflowException**
 (p. 773), **decaf::nio::BufferUnderflowException** (p. 776), **de-**
cafe::nio::InvalidMarkException (p. 1703), **decaf::nio::ReadOnlyBufferException**
 (p. 2522), **decaf::util::concurrent::BrokenBarrierException** (p. 685),
decaf::util::concurrent::CancellationException (p. 900), **de-**
cafe::util::concurrent::ExecutionException (p. 1509), **de-**
cafe::util::concurrent::RejectedExecutionException (p. 2535), and **de-**
cafe::util::concurrent::TimeoutException (p. 2989).

6.682.3.2 `virtual const std::exception* decaf::lang::Throwable::getCause () const` [pure virtual]

Gets the exception that caused this one to be thrown, this allows for chaining of exceptions in the case of a method that throws only a particular exception but wishes to allow for the real causal exception to be passed only in case the caller knows about that type of exception and wishes to respond to it.

Returns

a const pointer reference to the causal exception, if there was no cause associated with this exception then NULL is returned.

Implemented in **decaf::lang::Exception** (p. 1480).

6.682.3.3 `virtual std::string decaf::lang::Throwable::getMessage () const` [pure virtual]

Gets the cause of the error, if no message was provided to the instance of this interface but a cause was then the value `cause.getMessage` is then returned.

Returns

string errors message

Implemented in **decaf::lang::Exception** (p. 1480).

6.682.3.4 `virtual std::vector< std::pair< std::string, int> > decaf::lang::Throwable::getStackTrace () const` [pure virtual]

Provides the stack trace for every point where this exception was caught, marked, and rethrown.

Returns

vector containing stack trace strings

Implemented in **decaf::lang::Exception** (p. 1480).

6.682.3.5 **virtual std::string decaf::lang::Throwable::getStackTraceString () const**
[pure virtual]

Gets the stack trace as one contiguous string.

Returns

string with formatted stack trace data

Implemented in **decaf::lang::Exception** (p. 1481).

6.682.3.6 **virtual void decaf::lang::Throwable::initCause (const std::exception *
cause)** [pure virtual]

Initializes the contained cause exception with the one given.

A copy is made to avoid ownership issues.

Parameters

cause The exception that was the cause of this one.

Implemented in **decaf::lang::Exception** (p. 1481).

6.682.3.7 **virtual void decaf::lang::Throwable::printStackTrace () const** [pure
virtual]

Prints the stack trace to std::err.

Implemented in **decaf::lang::Exception** (p. 1481).

6.682.3.8 **virtual void decaf::lang::Throwable::printStackTrace (std::ostream &
stream) const** [pure virtual]

Prints the stack trace to the given output stream.

Parameters

stream the target output stream.

Implemented in **decaf::lang::Exception** (p. 1481).

6.682.3.9 **virtual void decaf::lang::Throwable::setMark (const char * file, const
int lineNumber)** [pure virtual]

Adds a file/line number to the stack trace.

Parameters

- file* The name of the file calling this method (use `__FILE__`).
- lineNumber* The line number in the calling file (use `__LINE__`).

Implemented in **decaf::lang::Exception** (p. 1482).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/Throwable.h`

6.683 decaf::util::concurrent::TimeoutException Class Reference

```
#include <src/main/decaf/util/concurrent/TimeoutException.h>
```

Inheritance diagram for `decaf::util::concurrent::TimeoutException`:

Public Member Functions

- **TimeoutException** () throw ()
Default Constructor.
- **TimeoutException** (const **decaf::lang::Exception** &ex) throw ()
Conversion Constructor from some other Exception.
- **TimeoutException** (const **TimeoutException** &ex) throw ()
Copy Constructor.
- **TimeoutException** (const std::exception *cause) throw ()
Constructor.
- **TimeoutException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **TimeoutException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **TimeoutException** * clone () const
Clones this exception.
- virtual ~**TimeoutException** () throw ()

6.683.1 Constructor & Destructor Documentation

6.683.1.1 decaf::util::concurrent::TimeoutException::TimeoutException () throw () [inline]

Default Constructor.

6.683.1.2 `decaf::util::concurrent::TimeoutException::TimeoutException (const decaf::lang::Exception & ex) throw () [inline]`

Conversion Constructor from some other Exception.

Parameters

ex An exception that should become this type of Exception

References `decaf::lang::Exception::Exception()`.

6.683.1.3 `decaf::util::concurrent::TimeoutException::TimeoutException (const TimeoutException & ex) throw () [inline]`

Copy Constructor.

Parameters

ex The exception to copy from.

References `decaf::lang::Exception::Exception()`.

6.683.1.4 `decaf::util::concurrent::TimeoutException::TimeoutException (const std::exception * cause) throw () [inline]`

Constructor.

Parameters

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.683.1.5 `decaf::util::concurrent::TimeoutException::TimeoutException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The string message to report

... list of primitives that are formatted into the message

6.683.1.6 `decaf::util::concurrent::TimeoutException::TimeoutException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw ()` [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The string message to report

... list of primitives that are formatted into the message

6.683.1.7 `virtual decaf::util::concurrent::TimeoutException::~~TimeoutException () throw ()` [inline, virtual]

6.683.2 Member Function Documentation

6.683.2.1 `virtual TimeoutException* decaf::util::concurrent::TimeoutException::clone () const` [inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

a new **TimeoutException** (p.2987) that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p.1479).

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/TimeoutException.h`

6.684 decaf::util::Timer Class Reference

A facility for threads to schedule tasks for future execution in a background thread.

```
#include <src/main/decaf/util/Timer.h>
```

Public Member Functions

- **Timer** ()
- virtual **~Timer** ()
- void **cancel** ()

Terminates this timer, discarding any currently scheduled tasks.

- `std::size_t purge ()`

Removes all canceled tasks from this timer's task queue.

- `void schedule (TimerTask *task, long long delay) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException)`

Schedules the specified task for execution after the specified delay.

- `void schedule (const decaf::lang::Pointer< TimerTask > &task, long long delay) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException)`

Schedules the specified task for execution after the specified delay.

- `void schedule (TimerTask *task, const Date &time) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException)`

Schedules the specified task for execution at the specified time.

- `void schedule (const decaf::lang::Pointer< TimerTask > &task, const Date &time) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException)`

Schedules the specified task for execution at the specified time.

- `void schedule (TimerTask *task, long long delay, long long period) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException)`

Schedules the specified task for repeated fixed-delay execution, beginning after the specified delay.

- `void schedule (const decaf::lang::Pointer< TimerTask > &task, long long delay, long long period) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException)`

Schedules the specified task for repeated fixed-delay execution, beginning after the specified delay.

- `void schedule (TimerTask *task, const Date &firstTime, long long period) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException)`

Schedules the specified task for repeated fixed-delay execution, beginning at the specified time.

- `void schedule (const decaf::lang::Pointer< TimerTask > &task, const Date &firstTime, long long period) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException)`

Schedules the specified task for repeated fixed-delay execution, beginning at the specified time.

- void **scheduleAtFixedRate** (**TimerTask** *task, long long delay, long long period) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException)

Schedules the specified task for repeated fixed-rate execution, beginning after the specified delay.

- void **scheduleAtFixedRate** (const decaf::lang::Pointer< **TimerTask** > &task, long long delay, long long period) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException)

Schedules the specified task for repeated fixed-rate execution, beginning after the specified delay.

- void **scheduleAtFixedRate** (**TimerTask** *task, const **Date** &firstTime, long long period) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException)

Schedules the specified task for repeated fixed-rate execution, beginning at the specified time.

- void **scheduleAtFixedRate** (const decaf::lang::Pointer< **TimerTask** > &task, const **Date** &firstTime, long long period) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException)

Schedules the specified task for repeated fixed-rate execution, beginning at the specified time.

6.684.1 Detailed Description

A facility for threads to schedule tasks for future execution in a background thread. Tasks may be scheduled for one-time execution, or for repeated execution at regular intervals.

Corresponding to each **Timer** (p. 2989) object is a single background thread that is used to execute all of the timer's tasks, sequentially. **Timer** (p. 2989) tasks should complete quickly. If a timer task takes excessive time to complete, it "hogs" the timer's task execution thread. This can, in turn, delay the execution of subsequent tasks, which may "bunch up" and execute in rapid succession when (and if) the offending task finally completes.

This class is thread-safe: multiple threads can share a single **Timer** (p. 2989) object without the need for external synchronization.

This class does not offer real-time guarantees: it schedules tasks using the wait(long) method.

Since

1.0

6.684.2 Constructor & Destructor Documentation

6.684.2.1 `decaf::util::Timer::Timer ()`

6.684.2.2 `virtual decaf::util::Timer::~~Timer () [virtual]`

6.684.3 Member Function Documentation

6.684.3.1 `void decaf::util::Timer::cancel ()`

Terminates this timer, discarding any currently scheduled tasks.

Does not interfere with a currently executing task (if it exists). Once a timer has been terminated, its execution thread terminates gracefully, and no more tasks may be scheduled on it.

Note that calling this method from within the run method of a timer task that was invoked by this timer absolutely guarantees that the ongoing task execution is the last task execution that will ever be performed by this timer.

This method may be called repeatedly; the second and subsequent calls have no effect.

6.684.3.2 `std::size_t decaf::util::Timer::purge ()`

Removes all canceled tasks from this timer's task queue.

Calling this method has no effect on the behavior of the timer, but eliminates the canceled tasks from the queue causing the **Timer** (p. 2989) to destroy the **TimerTask** (p. 3000) pointer it was originally given, the caller should ensure that they no longer have any references to TimerTasks that were previously scheduled.

Most programs will have no need to call this method. It is designed for use by the rare application that cancels a large number of tasks. Calling this method trades time for space: the runtime of the method may be proportional to $n + c \log n$, where n is the number of tasks in the queue and c is the number of canceled tasks.

This method can be called on a **Timer** (p. 2989) object that has no scheduled tasks without error.

Returns

the number of tasks removed from the queue.

6.684.3.3 `void decaf::util::Timer::schedule (const decaf::lang::Pointer< TimerTask > & task, long long delay) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException)`

Schedules the specified task for execution after the specified delay.

Parameters

task - task to be scheduled.

delay - delay in milliseconds before task is to be executed.

Exceptions

NullPointerException - if the **TimerTask** (p. 3000) value is Null.

IllegalArgumentException - if delay is negative, or delay + System.currentTimeMillis() is negative.

IllegalStateException - if task was already scheduled or cancelled, or timer was cancelled.

```
6.684.3.4 void decaf::util::Timer::schedule ( const decaf::lang::Pointer<
TimerTask > & task, long long delay, long long period
) throw ( decaf::lang::exceptions::NullPointerException,
decaf::lang::exceptions::IllegalArgumentException,
decaf::lang::exceptions::IllegalStateException )
```

Schedules the specified task for repeated fixed-delay execution, beginning after the specified delay.

Subsequent executions take place at approximately regular intervals separated by the specified period.

In fixed-delay execution, each execution is scheduled relative to the actual execution time of the previous execution. If an execution is delayed for any reason (such as other background activity), subsequent executions will be delayed as well. In the long run, the frequency of execution will generally be slightly lower than the reciprocal of the specified period (assuming the system clock underlying Object.wait(long long) is accurate).

Fixed-delay execution is appropriate for recurring activities that require "smoothness." In other words, it is appropriate for activities where it is more important to keep the frequency accurate in the short run than in the long run. This includes most animation tasks, such as blinking a cursor at regular intervals. It also includes tasks wherein regular activity is performed in response to human input, such as automatically repeating a character as long as a key is held down.

Parameters

task - task to be scheduled.

delay - delay in milliseconds before task is to be executed.

period - time in milliseconds between successive task executions.

Exceptions

NullPointerException - if the **TimerTask** (p. 3000) value is Null.

IllegalArgumentException - if delay is negative, or delay + System.currentTimeMillis() is negative.

IllegalStateException - if task was already scheduled or cancelled, timer was cancelled, or timer thread terminated.

```
6.684.3.5 void decaf::util::Timer::schedule ( TimerTask * task,
const Date & firstTime, long long period )
throw ( decaf::lang::exceptions::NullPointerException,
decaf::lang::exceptions::IllegalArgumentException,
decaf::lang::exceptions::IllegalStateException )
```

Schedules the specified task for repeated fixed-delay execution, beginning at the specified time.

Subsequent executions take place at approximately regular intervals separated by the specified period.

The **TimerTask** (p. 3000) pointer is considered to be owned by the **Timer** (p. 2989) class once it has been scheduled, the **Timer** (p. 2989) will destroy its **TimerTask**'s once they have been cancelled or the **Timer** (p. 2989) itself is cancelled. A **TimerTask** (p. 3000) is considered scheduled only when this method return without throwing an exception, until that time ownership is not considered to have been transferred to the **Timer** (p. 2989) and the caller should ensure that the **TimerTask** (p. 3000) gets deleted if an exception is thrown and no further attempts to schedule that **TimerTask** (p. 3000) instance are planned.

In fixed-delay execution, each execution is scheduled relative to the actual execution time of the previous execution. If an execution is delayed for any reason (such as other background activity), subsequent executions will be delayed as well. In the long run, the frequency of execution will generally be slightly lower than the reciprocal of the specified period (assuming the system clock underlying `Object.wait(long long)` is accurate).

Fixed-delay execution is appropriate for recurring activities that require "smoothness." In other words, it is appropriate for activities where it is more important to keep the frequency accurate in the short run than in the long run. This includes most animation tasks, such as blinking a cursor at regular intervals. It also includes tasks wherein regular activity is performed in response to human input, such as automatically repeating a character as long as a key is held down.

Parameters

task - task to be scheduled.

firstTime - First time at which task is to be executed.

period - time in milliseconds between successive task executions.

Exceptions

NullPointerException - if the **TimerTask** (p. 3000) value is Null.

IllegalArgumentException - if `time.getTime()` is negative.

IllegalStateException - if task was already scheduled or cancelled, timer was cancelled, or timer thread terminated.

```
6.684.3.6 void decaf::util::Timer::schedule ( TimerTask * task, const Date
& time ) throw ( decaf::lang::exceptions::NullPointerException,
decaf::lang::exceptions::IllegalArgumentException,
decaf::lang::exceptions::IllegalStateException )
```

Schedules the specified task for execution at the specified time.

If the time is in the past, the task is scheduled for immediate execution.

The **TimerTask** (p. 3000) pointer is considered to be owned by the **Timer** (p. 2989) class once it has been scheduled, the **Timer** (p. 2989) will destroy its **TimerTask**'s once they have been cancelled or the **Timer** (p. 2989) itself is cancelled. A **TimerTask** (p. 3000) is considered scheduled only when this method return without throwing an exception, until that time ownership is not considered to have been transferred to the **Timer** (p. 2989) and the caller should ensure that the **TimerTask** (p. 3000) gets deleted if an exception is thrown and no further attempts to schedule that **TimerTask** (p. 3000) instance are planned.

Parameters

task - task to be scheduled.

time - time at which task is to be executed.

Exceptions

NullPointerException - if the **TimerTask** (p. 3000) value is Null.

IllegalArgumentException - if time.getTime() is negative.

IllegalStateException - if task was already scheduled or cancelled, timer was cancelled, or timer thread terminated.

6.684.3.7 void decaf::util::Timer::schedule (const decaf::lang::Pointer< TimerTask > & *task*, const Date & *firstTime*, long long *period*) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException)

Schedules the specified task for repeated fixed-delay execution, beginning at the specified time.

Subsequent executions take place at approximately regular intervals separated by the specified period.

In fixed-delay execution, each execution is scheduled relative to the actual execution time of the previous execution. If an execution is delayed for any reason (such as other background activity), subsequent executions will be delayed as well. In the long run, the frequency of execution will generally be slightly lower than the reciprocal of the specified period (assuming the system clock underlying Object.wait(long long) is accurate).

Fixed-delay execution is appropriate for recurring activities that require "smoothness." In other words, it is appropriate for activities where it is more important to keep the frequency accurate in the short run than in the long run. This includes most animation tasks, such as blinking a cursor at regular intervals. It also includes tasks wherein regular activity is performed in response to human input, such as automatically repeating a character as long as a key is held down.

Parameters

task - task to be scheduled.

firstTime - First time at which task is to be executed.

period - time in milliseconds between successive task executions.

Exceptions

NullPointerException - if the **TimerTask** (p. 3000) value is Null.

IllegalArgumentException - if time.getTime() is negative.

IllegalStateException - if task was already scheduled or cancelled, timer was cancelled, or timer thread terminated.

6.684.3.8 void decaf::util::Timer::schedule (TimerTask * *task*, long long *delay*) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException)

Schedules the specified task for execution after the specified delay.

The **TimerTask** (p. 3000) pointer is considered to be owned by the **Timer** (p. 2989) class once it has been scheduled, the **Timer** (p. 2989) will destroy its TimerTask's once they have been cancelled or the **Timer** (p. 2989) itself is cancelled. A **TimerTask** (p. 3000) is considered scheduled

only when this method return without throwing an exception, until that time ownership is not considered to have been transferred to the **Timer** (p. 2989) and the caller should ensure that the **TimerTask** (p. 3000) gets deleted if an exception is thrown and no further attempts to schedule that **TimerTask** (p. 3000) instance are planned.

Parameters

task - task to be scheduled.

delay - delay in milliseconds before task is to be executed.

Exceptions

NullPointerException - if the **TimerTask** (p. 3000) value is Null.

IllegalArgumentException - if delay is negative, or delay + System.currentTimeMillis() is negative.

IllegalStateException - if task was already scheduled or cancelled, or timer was cancelled.

```
6.684.3.9 void decaf::util::Timer::schedule ( TimerTask * task, long long delay,
long long period ) throw ( decaf::lang::exceptions::NullPointerException,
decaf::lang::exceptions::IllegalArgumentException,
decaf::lang::exceptions::IllegalStateException )
```

Schedules the specified task for repeated fixed-delay execution, beginning after the specified delay.

Subsequent executions take place at approximately regular intervals separated by the specified period.

The **TimerTask** (p. 3000) pointer is considered to be owned by the **Timer** (p. 2989) class once it has been scheduled, the **Timer** (p. 2989) will destroy its **TimerTask**'s once they have been cancelled or the **Timer** (p. 2989) itself is cancelled. A **TimerTask** (p. 3000) is considered scheduled only when this method return without throwing an exception, until that time ownership is not considered to have been transferred to the **Timer** (p. 2989) and the caller should ensure that the **TimerTask** (p. 3000) gets deleted if an exception is thrown and no further attempts to schedule that **TimerTask** (p. 3000) instance are planned.

In fixed-delay execution, each execution is scheduled relative to the actual execution time of the previous execution. If an execution is delayed for any reason (such as other background activity), subsequent executions will be delayed as well. In the long run, the frequency of execution will generally be slightly lower than the reciprocal of the specified period (assuming the system clock underlying Object.wait(long long) is accurate).

Fixed-delay execution is appropriate for recurring activities that require "smoothness." In other words, it is appropriate for activities where it is more important to keep the frequency accurate in the short run than in the long run. This includes most animation tasks, such as blinking a cursor at regular intervals. It also includes tasks wherein regular activity is performed in response to human input, such as automatically repeating a character as long as a key is held down.

Parameters

task - task to be scheduled.

delay - delay in milliseconds before task is to be executed.

period - time in milliseconds between successive task executions.

Exceptions

NullPointerException - if the **TimerTask** (p. 3000) value is Null.

IllegalArgumentException - if delay is negative, or delay + System.currentTimeMillis() is negative.

IllegalStateException - if task was already scheduled or cancelled, timer was cancelled, or timer thread terminated.

```
6.684.3.10 void decaf::util::Timer::schedule ( const decaf::lang::Pointer<
TimerTask > & task, const Date & time )
throw ( decaf::lang::exceptions::NullPointerException,
decaf::lang::exceptions::IllegalArgumentException,
decaf::lang::exceptions::IllegalStateException )
```

Schedules the specified task for execution at the specified time.

If the time is in the past, the task is scheduled for immediate execution.

Parameters

task - task to be scheduled.

time - time at which task is to be executed.

Exceptions

NullPointerException - if the **TimerTask** (p. 3000) value is Null.

IllegalArgumentException - if time.getTime() is negative.

IllegalStateException - if task was already scheduled or cancelled, timer was cancelled, or timer thread terminated.

```
6.684.3.11 void decaf::util::Timer::scheduleAtFixedRate ( TimerTask
* task, long long delay, long long period )
throw ( decaf::lang::exceptions::NullPointerException,
decaf::lang::exceptions::IllegalArgumentException,
decaf::lang::exceptions::IllegalStateException )
```

Schedules the specified task for repeated fixed-rate execution, beginning after the specified delay.

Subsequent executions take place at approximately regular intervals, separated by the specified period.

The **TimerTask** (p. 3000) pointer is considered to be owned by the **Timer** (p. 2989) class once it has been scheduled, the **Timer** (p. 2989) will destroy its **TimerTask**'s once they have been cancelled or the **Timer** (p. 2989) itself is cancelled. A **TimerTask** (p. 3000) is considered scheduled only when this method return without throwing an exception, until that time ownership is not considered to have been transferred to the **Timer** (p. 2989) and the caller should ensure that the **TimerTask** (p. 3000) gets deleted if an exception is thrown and no further attempts to schedule that **TimerTask** (p. 3000) instance are planned.

In fixed-rate execution, each execution is scheduled relative to the scheduled execution time of the initial execution. If an execution is delayed for any reason (such as garbage collection or other background activity), two or more executions will occur in rapid succession to "catch up." In the long run, the frequency of execution will be exactly the reciprocal of the specified period (assuming the system clock underlying Object.wait(long) is accurate).

Fixed-rate execution is appropriate for recurring activities that are sensitive to absolute time, such as ringing a chime every hour on the hour, or running scheduled maintenance every day at

a particular time. It is also appropriate for recurring activities where the total time to perform a fixed number of executions is important, such as a countdown timer that ticks once every second for ten seconds. Finally, fixed-rate execution is appropriate for scheduling multiple repeating timer tasks that must remain synchronized with respect to one another.

Parameters

task - task to be scheduled.

delay - delay in milliseconds before task is to be executed.

period - time in milliseconds between successive task executions.

Exceptions

NullPointerException - if the **TimerTask** (p. 3000) value is Null.

IllegalArgumentException - if delay is negative, or delay + System.currentTimeMillis() is negative.

IllegalStateException - if task was already scheduled or cancelled, timer was cancelled, or timer thread terminated.

```
6.684.3.12 void decaf::util::Timer::scheduleAtFixedRate ( const
               decaf::lang::Pointer< TimerTask > & task, long long delay, long
               long period ) throw ( decaf::lang::exceptions::NullPointerException,
               decaf::lang::exceptions::IllegalArgumentException,
               decaf::lang::exceptions::IllegalStateException )
```

Schedules the specified task for repeated fixed-rate execution, beginning after the specified delay.

Subsequent executions take place at approximately regular intervals, separated by the specified period.

In fixed-rate execution, each execution is scheduled relative to the scheduled execution time of the initial execution. If an execution is delayed for any reason (such as garbage collection or other background activity), two or more executions will occur in rapid succession to "catch up." In the long run, the frequency of execution will be exactly the reciprocal of the specified period (assuming the system clock underlying Object.wait(long) is accurate).

Fixed-rate execution is appropriate for recurring activities that are sensitive to absolute time, such as ringing a chime every hour on the hour, or running scheduled maintenance every day at a particular time. It is also appropriate for recurring activities where the total time to perform a fixed number of executions is important, such as a countdown timer that ticks once every second for ten seconds. Finally, fixed-rate execution is appropriate for scheduling multiple repeating timer tasks that must remain synchronized with respect to one another.

Parameters

task - task to be scheduled.

delay - delay in milliseconds before task is to be executed.

period - time in milliseconds between successive task executions.

Exceptions

NullPointerException - if the **TimerTask** (p. 3000) value is Null.

IllegalArgumentException - if delay is negative, or delay + System.currentTimeMillis() is negative.

IllegalStateException - if task was already scheduled or cancelled, timer was cancelled, or timer thread terminated.

```
6.684.3.13 void decaf::util::Timer::scheduleAtFixedRate ( const
    decaf::lang::Pointer< TimerTask > & task, const
    Date & firstTime, long long period ) throw
    ( decaf::lang::exceptions::NullPointerException,
    decaf::lang::exceptions::IllegalArgumentException,
    decaf::lang::exceptions::IllegalStateException )
```

Schedules the specified task for repeated fixed-rate execution, beginning at the specified time.

Subsequent executions take place at approximately regular intervals, separated by the specified period.

In fixed-rate execution, each execution is scheduled relative to the scheduled execution time of the initial execution. If an execution is delayed for any reason (such as garbage collection or other background activity), two or more executions will occur in rapid succession to "catch up." In the long run, the frequency of execution will be exactly the reciprocal of the specified period (assuming the system clock underlying `Object.wait(long)` is accurate).

Fixed-rate execution is appropriate for recurring activities that are sensitive to absolute time, such as ringing a chime every hour on the hour, or running scheduled maintenance every day at a particular time. It is also appropriate for recurring activities where the total time to perform a fixed number of executions is important, such as a countdown timer that ticks once every second for ten seconds. Finally, fixed-rate execution is appropriate for scheduling multiple repeating timer tasks that must remain synchronized with respect to one another.

Parameters

task - task to be scheduled.

firstTime - First time at which task is to be executed.

period - time in milliseconds between successive task executions.

Exceptions

NullPointerException - if the **TimerTask** (p. 3000) value is Null.

IllegalArgumentException - if `time.getTime()` is negative.

IllegalStateException - if task was already scheduled or cancelled, timer was cancelled, or timer thread terminated.

```
6.684.3.14 void decaf::util::Timer::scheduleAtFixedRate ( TimerTask
    * task, const Date & firstTime, long long period )
    throw ( decaf::lang::exceptions::NullPointerException,
    decaf::lang::exceptions::IllegalArgumentException,
    decaf::lang::exceptions::IllegalStateException )
```

Schedules the specified task for repeated fixed-rate execution, beginning at the specified time.

Subsequent executions take place at approximately regular intervals, separated by the specified period.

The **TimerTask** (p. 3000) pointer is considered to be owned by the **Timer** (p. 2989) class once it has been scheduled, the **Timer** (p. 2989) will destroy its **TimerTask**'s once they have been cancelled or the **Timer** (p. 2989) itself is cancelled. A **TimerTask** (p. 3000) is considered scheduled only when this method return without throwing an exception, until that time ownership is not considered to have been transferred to the **Timer** (p. 2989) and the caller should ensure that the **TimerTask** (p. 3000) gets deleted if an exception is thrown and no further attempts to schedule that **TimerTask** (p. 3000) instance are planned.

In fixed-rate execution, each execution is scheduled relative to the scheduled execution time of the initial execution. If an execution is delayed for any reason (such as garbage collection or other background activity), two or more executions will occur in rapid succession to "catch up." In the long run, the frequency of execution will be exactly the reciprocal of the specified period (assuming the system clock underlying `Object.wait(long)` is accurate).

Fixed-rate execution is appropriate for recurring activities that are sensitive to absolute time, such as ringing a chime every hour on the hour, or running scheduled maintenance every day at a particular time. It is also appropriate for recurring activities where the total time to perform a fixed number of executions is important, such as a countdown timer that ticks once every second for ten seconds. Finally, fixed-rate execution is appropriate for scheduling multiple repeating timer tasks that must remain synchronized with respect to one another.

Parameters

task - task to be scheduled.

firstTime - First time at which task is to be executed.

period - time in milliseconds between successive task executions.

Exceptions

NullPointerException - if the **TimerTask** (p. 3000) value is Null.

IllegalArgumentException - if `time.getTime()` is negative.

IllegalStateException - if task was already scheduled or cancelled, timer was cancelled, or timer thread terminated.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/Timer.h`

6.685 decaf::util::TimerTask Class Reference

A Base class for a task object that can be scheduled for one-time or repeated execution by a **Timer** (p. 2989).

```
#include <src/main/decaf/util/TimerTask.h>
```

Inheritance diagram for `decaf::util::TimerTask`:

Public Member Functions

- **TimerTask** ()
- virtual **~TimerTask** ()

- bool **cancel** ()
Cancels this timer task.
- long long **scheduledExecutionTime** () const
Returns the scheduled execution time of the most recent actual execution of this task.

Protected Member Functions

- bool **isScheduled** () const
- void **setScheduledTime** (long long time)
- long long **getWhen** () const

Friends

- class **Timer**
- class **TimerImpl**
- class **decaf::internal::util::TimerTaskHeap**

6.685.1 Detailed Description

A Base class for a task object that can be scheduled for one-time or repeated execution by a **Timer** (p. 2989).

Since

1.0

6.685.2 Constructor & Destructor Documentation

6.685.2.1 decaf::util::TimerTask::TimerTask ()

6.685.2.2 virtual decaf::util::TimerTask::~~TimerTask () [inline, virtual]

6.685.3 Member Function Documentation

6.685.3.1 bool decaf::util::TimerTask::cancel ()

Cancels this timer task.

If the task has been scheduled for one-time execution and has not yet run, or has not yet been scheduled, it will never run. If the task has been scheduled for repeated execution, it will never run again. (If the task is running when this call occurs, the task will run to completion, but will never run again.)

Note that calling this method from within the run method of a repeating timer task absolutely guarantees that the timer task will not run again.

This method may be called repeatedly; the second and subsequent calls have no effect.

Returns

true if this task is scheduled for one-time execution and has not yet run, or this task is scheduled for repeated execution. Returns false if the task was scheduled for one-time execution and has already run, or if the task was never scheduled, or if the task was already canceled. (Loosely speaking, this method returns true if it prevents one or more scheduled executions from taking place.)

6.685.3.2 `long long decaf::util::TimerTask::getWhen () const` [protected]

6.685.3.3 `bool decaf::util::TimerTask::isScheduled () const` [protected]

6.685.3.4 `long long decaf::util::TimerTask::scheduledExecutionTime () const`

Returns the scheduled execution time of the most recent actual execution of this task.

(If this method is invoked while task execution is in progress, the return value is the scheduled execution time of the ongoing task execution.)

This method is typically invoked from within a task's run method, to determine whether the current execution of the task is sufficiently timely to warrant performing the scheduled activity:

```
void run() (p. 2642) { if( System::currentTimeMillis() - scheduledExecutionTime() (p. 3002)
>= MAX_TARDINESS) return; // Too late; skip this execution. // Perform the task }
```

This method is typically not used in conjunction with fixed-delay execution repeating tasks, as their scheduled execution times are allowed to drift over time, and so are not terribly significant.

Returns

the time at which the most recent execution of this task was scheduled to occur, in the format returned by **Date.getTime()** (p. 1380). The return value is undefined if the task has yet to commence its first execution.

6.685.3.5 `void decaf::util::TimerTask::setScheduledTime (long long time)`
[protected]

6.685.4 Friends And Related Function Documentation

6.685.4.1 `friend class decaf::internal::util::TimerTaskHeap` [friend]

6.685.4.2 `friend class Timer` [friend]

6.685.4.3 `friend class TimerImpl` [friend]

The documentation for this class was generated from the following file:

- `src/main/decaf/util/TimerTask.h`

6.686 decaf::internal::util::TimerTaskHeap Class Reference

A Binary Heap implemented specifically for the Timer class in Decaf Util.

```
#include <src/main/decaf/internal/util/TimerTaskHeap.h>
```

Public Member Functions

- **TimerTaskHeap** ()
- virtual **~TimerTaskHeap** ()
- **Pointer< TimerTask > peek** ()
Peeks at the Head of the Heap, returns the task with the nearest scheduled run time.
- bool **isEmpty** () const
- std::size_t **size** () const
- void **insert** (const **Pointer< TimerTask >** &task)
Inserts the specified Task into the heap, heap is reordered to reflect the addition of a new element.
- void **remove** (std::size_t pos)
Removes the Task at the specified position from the heap, resorts the heap from the position down to the bottom.
- void **reset** ()
Clear all contents from the heap.
- void **adjustMinimum** ()
Resorts the heap starting at the top.
- std::size_t **deleteIfCancelled** ()
Runs through the heap removing all cancelled Tasks from it, this is not normally used but in case a cancellation of a large number of tasks the user can perform this purge.
- std::size_t **find** (const **Pointer< TimerTask >** &task) const
Searches the heap for the specified TimerTask element and returns its position in the heap.

6.686.1 Detailed Description

A Binary Heap implemented specifically for the Timer class in Decaf Util.

Since

1.0

6.686.2 Constructor & Destructor Documentation

6.686.2.1 decaf::internal::util::TimerTaskHeap::TimerTaskHeap ()

6.686.2.2 virtual decaf::internal::util::TimerTaskHeap::~~TimerTaskHeap ()
 [virtual]

6.686.3 Member Function Documentation

6.686.3.1 void decaf::internal::util::TimerTaskHeap::adjustMinimum ()

Resorts the heap starting at the top.

6.686.3.2 `std::size_t decaf::internal::util::TimerTaskHeap::deleteIfCancelled ()`

Runs through the heap removing all cancelled Tasks from it, this is not normally used but in case a cancellation of a large number of tasks the user can perform this purge.

Returns

the number of task that were removed from the heap because they were cancelled.

6.686.3.3 `std::size_t decaf::internal::util::TimerTaskHeap::find (const Pointer< TimerTask > & task) const`

Searches the heap for the specified TimerTask element and returns its position in the heap.

Returns the unsigned equivalent of -1 if the element is not found.

Returns

the position in the Heap where the Task is stored, or npos.

6.686.3.4 `void decaf::internal::util::TimerTaskHeap::insert (const Pointer< TimerTask > & task)`

Inserts the specified Task into the heap, heap is reordered to reflect the addition of a new element.

Parameters

task The TimerTask to insert into the heap.

6.686.3.5 `bool decaf::internal::util::TimerTaskHeap::isEmpty () const`**Returns**

true if the heap is empty.

6.686.3.6 `Pointer<TimerTask> decaf::internal::util::TimerTaskHeap::peek ()`

Peeks at the Head of the Heap, returns the task with the nearest scheduled run time.

Returns

The TimerTask that is scheduled to be executed next if the Heap is empty a Null Pointer value is returned.

6.686.3.7 `void decaf::internal::util::TimerTaskHeap::remove (std::size_t pos)`

Removes the Task at the specified position from the heap, resorts the heap from the position down to the bottom.

Parameters

pos The position at which to remove the TimerTask and begin a resort of the heap.

6.686.3.8 void decaf::internal::util::TimerTaskHeap::reset ()

Clear all contents from the heap.

6.686.3.9 std::size_t decaf::internal::util::TimerTaskHeap::size () const

Returns

the size of the heap.

The documentation for this class was generated from the following file:

- src/main/decaf/internal/util/TimerTaskHeap.h

6.687 decaf::util::concurrent::TimeUnit Class Reference

A **TimeUnit** (p. 3005) represents time durations at a given unit of granularity and provides utility methods to convert across units, and to perform timing and delay operations in these units.

```
#include <src/main/decaf/util/concurrent/TimeUnit.h>
```

Inheritance diagram for decaf::util::concurrent::TimeUnit:

Public Member Functions

- virtual **~TimeUnit** ()
- long long **convert** (long long sourceDuration, const **TimeUnit** &sourceUnit) const
Convert the given time duration in the given unit to this unit.
- long long **toNanos** (long long duration) const
Equivalent to `NANOSECONDS.convert(duration, this)`.
- long long **toMicros** (long long duration) const
Equivalent to `MICROSECONDS.convert(duration, this)`.
- long long **toMillis** (long long duration) const
Equivalent to `MILLISECONDS.convert(duration, this)`.
- long long **toSeconds** (long long duration) const
Equivalent to `SECONDS.convert(duration, this)`.
- long long **toMinutes** (long long duration) const
Equivalent to `MINUTES.convert(duration, this)`.
- long long **toHours** (long long duration) const
Equivalent to `HOURS.convert(duration, this)`.
- long long **toDays** (long long duration) const

Equivalent to `DAYS.convert(duration, this)`.

- void **timedWait** (**Synchronizable** *obj, long long timeout) const throw (decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::NullPointerException)
Perform a timed `Object.wait` using this time unit.
- void **timedJoin** (decaf::lang::Thread *thread, long long timeout) throw (decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::NullPointerException)
Perform a timed `Thread.join` using this time unit.
- void **sleep** (long long timeout) const throw (decaf::lang::exceptions::InterruptedException)
Perform a `Thread.sleep` using this unit.
- virtual std::string **toString** () const
*Converts the **TimeUnit** (p. 3005) type to the Name of the **TimeUnit** (p. 3005).*
- virtual int **compareTo** (const **TimeUnit** &value) const
Compares this object with the specified object for order.
- virtual bool **equals** (const **TimeUnit** &value) const
- virtual bool **operator==** (const **TimeUnit** &value) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const **TimeUnit** &value) const
Compares this object to another and returns true if this object is considered to be less than the one passed.

Static Public Member Functions

- static const **TimeUnit** & **valueOf** (const std::string &name) throw (decaf::lang::exceptions::IllegalArgumentException)
*Returns the **TimeUnit** (p. 3005) constant of this type with the specified name.*

Static Public Attributes

- static const **TimeUnit** **NANOSECONDS**
*The Actual **TimeUnit** (p. 3005) enumerations.*
- static const **TimeUnit** **MICROSECONDS**
- static const **TimeUnit** **MILLISECONDS**
- static const **TimeUnit** **SECONDS**
- static const **TimeUnit** **MINUTES**
- static const **TimeUnit** **HOURS**
- static const **TimeUnit** **DAYS**
- static const **TimeUnit** *const **values** []
*The An Array of **TimeUnit** (p. 3005) Instances.*

Protected Member Functions

- **TimeUnit** (int index, const std::string &name)

Hidden Constructor, this class can not be instantiated directly.

6.687.1 Detailed Description

A **TimeUnit** (p. 3005) represents time durations at a given unit of granularity and provides utility methods to convert across units, and to perform timing and delay operations in these units. A **TimeUnit** (p. 3005) does not maintain time information, but only helps organize and use time representations that may be maintained separately across various contexts. A nanosecond is defined as one thousandth of a microsecond, a microsecond as one thousandth of a millisecond, a millisecond as one thousandth of a second, a minute as sixty seconds, an hour as sixty minutes, and a day as twenty four hours.

A **TimeUnit** (p. 3005) is mainly used to inform time-based methods how a given timing parameter should be interpreted. For example, the following code will timeout in 50 milliseconds if the lock is not available:

```
Lock (p. 1903) lock = ...; if ( lock.tryLock( 50, TimeUnit::MILLISECONDS (p. 3013) ) ) ...
```

while this code will timeout in 50 seconds:

```
Lock (p. 1903) lock = ...; if ( lock.tryLock( 50, TimeUnit::SECONDS (p. 3014) ) ) ...
```

Note however, that there is no guarantee that a particular timeout implementation will be able to notice the passage of time at the same granularity as the given **TimeUnit** (p. 3005).

6.687.2 Constructor & Destructor Documentation

6.687.2.1 `decaf::util::concurrent::TimeUnit::TimeUnit (int index, const std::string & name)` [protected]

Hidden Constructor, this class can not be instantiated directly.

Parameters

index - Index into the Time Unit set.

name - Name of the unit type being represented.

6.687.2.2 `virtual decaf::util::concurrent::TimeUnit::~~TimeUnit ()` [inline, virtual]

6.687.3 Member Function Documentation

6.687.3.1 `virtual int decaf::util::concurrent::TimeUnit::compareTo (const TimeUnit & value) const` [virtual]

Compares this object with the specified object for order.

Returns a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

In the foregoing description, the notation `sgn(expression)` designates the mathematical signum function, which is defined to return one of -1, 0, or 1 according to whether the value of expression is negative, zero or positive. The implementor must ensure `sgn(x.compareTo(y)) == -sgn(y.compareTo(x))` for all `x` and `y`. (This implies that `x.compareTo(y)` must throw an exception iff `y.compareTo(x)` throws an exception.)

The implementor must also ensure that the relation is transitive: `(x.compareTo(y)>0 && y.compareTo(z)>0)` implies `x.compareTo(z)>0`.

Finally, the implementor must ensure that `x.compareTo(y)==0` implies that `sgn(x.compareTo(z)) == sgn(y.compareTo(z))`, for all `z`.

It is strongly recommended, but not strictly required that `(x.compareTo(y)==0) == (x.equals(y))`. Generally speaking, any class that implements the `Comparable` interface and violates this condition should clearly indicate this fact. The recommended language is "Note: this class has a natural ordering that is inconsistent with equals."

Parameters

value - the Object to be compared.

Returns

a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

6.687.3.2 `long long decaf::util::concurrent::TimeUnit::convert (long long sourceDuration, const TimeUnit & sourceUnit) const`

Convert the given time duration in the given unit to this unit.

Conversions from finer to coarser granularities truncate, so lose precision. For example converting 999 milliseconds to seconds results in 0. Conversions from coarser to finer granularities with arguments that would numerically overflow saturate to `Long.MIN_VALUE` if negative or `Long.MAX_VALUE` if positive.

For example, to convert 10 minutes to milliseconds, use: `TimeUnit.MILLISECONDS.convert(10L, TimeUnit.MINUTES)` (p. 3013))

Parameters

sourceDuration - Duration value to convert.

sourceUnit - Unit type of the source duration.

Returns

the converted duration in this unit, or `Long.MIN_VALUE` if conversion would negatively overflow, or `Long.MAX_VALUE` if it would positively overflow.

6.687.3.3 `virtual bool decaf::util::concurrent::TimeUnit::equals (const TimeUnit & value) const [virtual]`

Returns

true if this value is considered equal to the passed value.

6.687.3.4 virtual bool decaf::util::concurrent::TimeUnit::operator< (const TimeUnit & *value*) const [virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

Parameters

value - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

6.687.3.5 virtual bool decaf::util::concurrent::TimeUnit::operator== (const TimeUnit & *value*) const [virtual]

Compares equality between this object and the one passed.

Parameters

value - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

6.687.3.6 void decaf::util::concurrent::TimeUnit::sleep (long long *timeout*) const throw (decaf::lang::exceptions::InterruptedException)

Perform a Thread.sleep using this unit.

This is a convenience method that converts time arguments into the form required by the Thread.sleep method.

Parameters

timeout the minimum time to sleep

See also

Thread::sleep

6.687.3.7 void decaf::util::concurrent::TimeUnit::timedJoin (decaf::lang::Thread * *thread*, long long *timeout*) throw (decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::NullPointerException)

Perform a timed Thread.join using this time unit.

This is a convenience method that converts time arguments into the form required by the Thread.join method.

Parameters

thread the thread to wait for

timeout the maximum time to wait

Exceptions

InterruptedException if interrupted while waiting.

NullPointerException if the thread object is null.

See also

Thread::join(long long, long long)

6.687.3.8 `void decaf::util::concurrent::TimeUnit::timedWait (Synchronizable * obj, long long timeout) const throw (decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::NullPointerException)`

Perform a timed `Object.wait` using this time unit.

This is a convenience method that converts timeout arguments into the form required by the `Object.wait` method.

For example, you could implement a blocking poll method (see `BlockingQueue.poll` (p. ??)) using:

```
Object poll( long long timeout, const TimeUnit& unit )
    throw( InterruptedException ) {

    while( empty ) {
        unit.timedWait(this, timeout);
        ...
    }
}
```

Parameters

obj the object to wait on

timeout the maximum time to wait.

Exceptions

InterruptedException if interrupted while waiting.

NullPointerException if the `Synchronizable` (p. 2930) object is null.

See also

Synchronizable::wait(long long, long long)

6.687.3.9 `long long decaf::util::concurrent::TimeUnit::toDays (long long duration) const [inline]`

Equivalent to `DAYS.convert(duration, this)`.

Parameters

duration the duration

Returns

the converted duration.

See also

`convert` (p. 3008)

6.687.3.10 `long long decaf::util::concurrent::TimeUnit::toHours (long long duration) const [inline]`

Equivalent to `HOURS.convert(duration, this)`.

Parameters

duration the duration

Returns

the converted duration.

See also

`convert` (p. 3008)

6.687.3.11 `long long decaf::util::concurrent::TimeUnit::toMicros (long long duration) const [inline]`

Equivalent to `MICROSECONDS.convert(duration, this)`.

Parameters

duration the duration

Returns

the converted duration, or `Long.MIN_VALUE` if conversion would negatively overflow, or `Long.MAX_VALUE` if it would positively overflow.

See also

`convert` (p. 3008)

6.687.3.12 `long long decaf::util::concurrent::TimeUnit::toMillis (long long duration) const [inline]`

Equivalent to `MILLISECONDS.convert(duration, this)`.

Parameters

duration the duration

Returns

the converted duration, or `Long.MIN_VALUE` if conversion would negatively overflow, or `Long.MAX_VALUE` if it would positively overflow.

See also

`convert` (p. 3008)

6.687.3.13 `long long decaf::util::concurrent::TimeUnit::toMinutes (long long duration) const [inline]`

Equivalent to `MINUTES.convert(duration, this)`.

Parameters

duration the duration

Returns

the converted duration.

See also

`convert` (p. 3008)

6.687.3.14 `long long decaf::util::concurrent::TimeUnit::toNanos (long long duration) const [inline]`

Equivalent to `NANOSECONDS.convert(duration, this)`.

Parameters

duration the duration

Returns

the converted duration, or `Long.MIN_VALUE` if conversion would negatively overflow, or `Long.MAX_VALUE` if it would positively overflow.

See also

`convert` (p. 3008)

6.687.3.15 `long long decaf::util::concurrent::TimeUnit::toSeconds (long long duration) const [inline]`

Equivalent to `SECONDS.convert(duration, this)`.

Parameters

duration the duration

Returns

the converted duration.

See also

`convert` (p. 3008)

6.687.3.16 `virtual std::string decaf::util::concurrent::TimeUnit::toString () const`
[virtual]

Converts the **TimeUnit** (p. 3005) type to the Name of the **TimeUnit** (p. 3005).

Returns

String name of the **TimeUnit** (p. 3005)

6.687.3.17 `static const TimeUnit& decaf::util::concurrent::TimeUnit::valueOf`
(`const std::string & name`) `throw (de-`
`cafe::lang::exceptions::IllegalArgumentException)`
[static]

Returns the **TimeUnit** (p. 3005) constant of this type with the specified name.

The string must match exactly an identifier used to declare an **TimeUnit** (p. 3005) constant in this type. (Extraneous whitespace characters are not permitted.)

Parameters

name The Name of the **TimeUnit** (p. 3005) constant to be returned.

Returns

A constant reference to the **TimeUnit** (p. 3005) Constant with the given name.

Exceptions

IllegalArgumentException if this enum type has no constant with the specified name

6.687.4 Field Documentation

6.687.4.1 `const TimeUnit decaf::util::concurrent::TimeUnit::DAYS` [static]

6.687.4.2 `const TimeUnit decaf::util::concurrent::TimeUnit::HOURS` [static]

6.687.4.3 `const TimeUnit decaf::util::concurrent::TimeUnit::MICROSECONDS`
[static]

6.687.4.4 `const TimeUnit decaf::util::concurrent::TimeUnit::MILLISECONDS`
[static]

6.687.4.5 `const TimeUnit decaf::util::concurrent::TimeUnit::MINUTES` [static]

6.687.4.6 `const TimeUnit decaf::util::concurrent::TimeUnit::NANOSECONDS`
[static]

The Actual **TimeUnit** (p. 3005) enumerations.

6.687.4.7 `const TimeUnit decaf::util::concurrent::TimeUnit::SECONDS` [static]

6.687.4.8 `const TimeUnit* const decaf::util::concurrent::TimeUnit::values[]`
[static]

The An Array of **TimeUnit** (p. 3005) Instances.

The documentation for this class was generated from the following file:

- `src/main/decaf/util/concurrent/TimeUnit.h`

6.688 cms::Topic Class Reference

An interface encapsulating a provider-specific topic name.

```
#include <src/main/cms/Topic.h>
```

Inheritance diagram for cms::Topic:

Public Member Functions

- virtual `~Topic ()`
- virtual `std::string getTopicName () const =0 throw (CMSEException)`
Gets the name of this topic.

6.688.1 Detailed Description

An interface encapsulating a provider-specific topic name. A **Topic** (p.3014) is a Publish / Subscribe type **Destination** (p.1387). All Messages sent to a **Topic** (p.3014) are broadcast to all Subscribers of that **Topic** (p.3014) unless the Subscriber defines a **Message** (p.2036) selector that filters out that **Message** (p.2036).

Since

1.0

6.688.2 Constructor & Destructor Documentation

6.688.2.1 `virtual cms::Topic::~~Topic ()` [inline, virtual]

6.688.3 Member Function Documentation

6.688.3.1 `virtual std::string cms::Topic::getTopicName () const throw (CMSEException)` [pure virtual]

Gets the name of this topic.

Returns

The topic name.

Exceptions

CMSException (p. 960) - If an internal error occurs.

Implemented in **activemq::commands::ActiveMQTopic** (p. 553).

The documentation for this class was generated from the following file:

- `src/main/cms/Topic.h`

6.689 activemq::state::Tracked Class Reference

```
#include <src/main/activemq/state/Tracked.h>
```

Inheritance diagram for `activemq::state::Tracked`:

Public Member Functions

- **Tracked** ()
- **Tracked** (const **Pointer**< **decaf::lang::Runnable** > &runnable)
- virtual **~Tracked** ()
- void **onResponse** ()
- bool **isWaitingForResponse** () const

6.689.1 Constructor & Destructor Documentation

6.689.1.1 `activemq::state::Tracked::Tracked ()` [inline]

6.689.1.2 `activemq::state::Tracked::Tracked (const Pointer< decaf::lang::Runnable > & runnable)`

6.689.1.3 `virtual activemq::state::Tracked::~~Tracked ()` [inline, virtual]

6.689.2 Member Function Documentation

6.689.2.1 `bool activemq::state::Tracked::isWaitingForResponse ()` const [inline]

6.689.2.2 `void activemq::state::Tracked::onResponse ()`

The documentation for this class was generated from the following file:

- `src/main/activemq/state/Tracked.h`

6.690 activemq::commands::TransactionId Class Reference

```
#include <src/main/activemq/commands/TransactionId.h>
```

Inheritance diagram for `activemq::commands::TransactionId`:

Public Types

- `typedef decaf::lang::PointerComparator< TransactionId > COMPARATOR`

Public Member Functions

- `TransactionId ()`
- `TransactionId (const TransactionId &other)`
- `virtual ~TransactionId ()`
- `virtual unsigned char getDataStructureType () const`
Get the unique identifier that this object and its own Marshaler share.
- `virtual TransactionId * cloneDataStructure () const`
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- `virtual void copyDataStructure (const DataStructure *src)`
Copy the contents of the passed object into this object's members, overwriting any existing data.
- `virtual std::string toString () const`
*Returns a string containing the information for this **DataStructure** (p. 1372) such as its type and value of its elements.*
- `virtual bool equals (const DataStructure *value) const`
*Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent.*
- `virtual int compareTo (const TransactionId &value) const`
- `virtual bool equals (const TransactionId &value) const`
- `virtual bool operator== (const TransactionId &value) const`
- `virtual bool operator< (const TransactionId &value) const`
- `TransactionId & operator= (const TransactionId &other)`

Static Public Attributes

- `static const unsigned char ID_TRANSACTIONID = 0`

6.690.1 Member Typedef Documentation

6.690.1.1 `typedef decaf::lang::PointerComparator<TransactionId>
activemq::commands::TransactionId::COMPARATOR`

6.690.2 Constructor & Destructor Documentation

6.690.2.1 `activemq::commands::TransactionId::TransactionId ()`

6.690.2.2 `activemq::commands::TransactionId::TransactionId (const
TransactionId & other)`

6.690.2.3 `virtual activemq::commands::TransactionId::~~TransactionId ()
[virtual]`

6.690.3 Member Function Documentation

6.690.3.1 `virtual TransactionId* ac-
tivemq::commands::TransactionId::cloneDataStructure (
) const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

6.690.3.2 `virtual int activemq::commands::TransactionId::compareTo (const
TransactionId & value) const [virtual]`

6.690.3.3 `virtual void activemq::commands::TransactionId::copyDataStructure (
const DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

6.690.3.4 `virtual bool activemq::commands::TransactionId::equals (const
DataStructure * value) const [virtual]`

Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

6.690.3.5 `virtual bool activemq::commands::TransactionId::equals (const TransactionId & value) const` [virtual]

6.690.3.6 `virtual unsigned char activemq::commands::TransactionId::getDataStructureType () const` [virtual]

Get the unique identifier that this object and its own Marshaller share.

Returns

new **DataStructure** (p. 1372) type copy.

6.690.3.7 `virtual bool activemq::commands::TransactionId::operator< (const TransactionId & value) const` [virtual]

6.690.3.8 `TransactionId& activemq::commands::TransactionId::operator= (const TransactionId & other)`

6.690.3.9 `virtual bool activemq::commands::TransactionId::operator== (const TransactionId & value) const` [virtual]

6.690.3.10 `virtual std::string activemq::commands::TransactionId::toString () const` [virtual]

Returns a string containing the information for this **DataStructure** (p. 1372) such as its type and value of its elements.

Returns

formatted string useful for debugging.

6.690.4 Field Documentation

6.690.4.1 `const unsigned char activemq::commands::TransactionId::ID_TRANSACTIONID = 0` [static]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/TransactionId.h`

6.691 activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3018).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/TransactionIdMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller`:

Public Member Functions

- **TransactionIdMarshaller** ()
- virtual **~TransactionIdMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

6.691.1 Detailed Description

Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3018). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.691.2 Constructor & Destructor Documentation

6.691.2.1 `activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller::TransactionIdMarshaller () [inline]`

6.691.2.2 `virtual
activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller::~~TransactionIdMarshaller () [inline, virtual]`

6.691.3 Member Function Documentation

6.691.3.1 `virtual void activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1342).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller` (p. 1880), and `activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller` (p. 3199).

6.691.3.2 `virtual void activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1348).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller** (p. 1880), and **activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller** (p. 3199).

6.691.3.3 `virtual int activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1354).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller** (p. 1880), and **activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller** (p. 3199).

6.691.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller::tightMarshal2 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1360).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller` (p. 1881), and `activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller` (p. 3200).

6.691.3.5 `virtual void activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]`

Un-marshall an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1366).

Reimplemented in `activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller` (p. 1881), and `activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller` (p. 3200).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/TransactionIdMarshaller.h`

6.692 `activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller` Class Reference

Marshaling code for Open Wire Format for `TransactionIdMarshaller` (p. 3022).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/TransactionIdMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller`:

Public Member Functions

- `TransactionIdMarshaller ()`
- `virtual ~TransactionIdMarshaller ()`
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.692.1 Detailed Description

Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3022). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.692.2 Constructor & Destructor Documentation

6.692.2.1 activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller::TransactionIdMarshaller () [inline]

6.692.2.2 virtual activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller::~~TransactionIdMarshaller () [inline, virtual]

6.692.3 Member Function Documentation

6.692.3.1 virtual void activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1342).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller` (p. 1892), and `activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller` (p. 3211).

6.692.3.2 `virtual void activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1348).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller` (p. 1892), and `activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller` (p. 3211).

6.692.3.3 `virtual int activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1354).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller` (p. 1892), and `activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller` (p. 3211).

6.692.3.4 virtual void `activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller::tightMarshal2`
(`OpenWireFormat * wireFormat`, `commands::DataStructure`
* `dataStructure`, `decaf::io::DataOutputStream * dataOut`,
`utils::BooleanStream * bs`) throw (`decaf::io::IOException`) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1360).

Reimplemented in `activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller` (p. 1893), and `activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller` (p. 3212).

6.692.3.5 virtual void `activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller::tightUnmarshal`
(`OpenWireFormat * wireFormat`, `commands::DataStructure`
* `dataStructure`, `decaf::io::DataInputStream * dataIn`,
`utils::BooleanStream * bs`) throw (`decaf::io::IOException`) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1366).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller** (p. 1893), and **activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller** (p. 3212).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/TransactionIdMarshaller.h`

6.693 **activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller** Class Reference

Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3026).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/TransactionIdMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller**:

Public Member Functions

- **TransactionIdMarshaller** ()
- virtual **~TransactionIdMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.693.1 Detailed Description

Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3026). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.693.2 Constructor & Destructor Documentation

6.693.2.1 `activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller::TransactionIdMarshaller ()` [inline]

6.693.2.2 `virtual activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller::~~TransactionIdMarshaller ()` [inline, virtual]

6.693.3 Member Function Documentation

6.693.3.1 `virtual void activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1342).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller** (p. 1888), and **activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller** (p. 3207).

6.693.3.2 `virtual void activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1348).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller` (p. 1888), and `activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller` (p. 3207).

```
6.693.3.3 virtual int ac-
tivemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller::tightMarshal1
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, utils::BooleanStream * bs ) throw (
decaf::io::IOException ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1354).

Reimplemented in `activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller` (p. 1888), and `activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller` (p. 3207).

```
6.693.3.4 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1360).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller** (p. 1889), and **activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller** (p. 3208).

6.693.3.5 virtual void activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller::tightUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn, utils::BooleanStream * bs) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1366).

Reimplemented in **activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller** (p. 1889), and **activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller** (p. 3208).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/TransactionIdMarshaller.h`

6.694 activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3029).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/TransactionIdMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller**:

Public Member Functions

- **TransactionIdMarshaller** ()
- virtual **~TransactionIdMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Un-marshall an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshall an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)

Write a object instance to data output stream.

6.694.1 Detailed Description

Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3029). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.694.2 Constructor & Destructor Documentation

6.694.2.1 `activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller::TransactionIdMarshaller () [inline]`

6.694.2.2 `virtual
activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller::~~TransactionIdMarshaller () [inline, virtual]`

6.694.3 Member Function Documentation

6.694.3.1 `virtual void activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1342).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller` (p. 1896), and `activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller` (p. 3215).

6.694.3.2 `virtual void activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException) [virtual]`

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1348).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller` (p. 1896), and `activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller` (p. 3215).

```
6.694.3.3  virtual int ac-
            tivemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller::tightMarshal1
            ( OpenWireFormat * wireFormat, commands::DataStructure
            * dataStructure, utils::BooleanStream * bs ) throw (
            decaf::io::IOException ) [virtual]
```

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1354).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller` (p. 1896), and `activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller` (p. 3215).

```
6.694.3.4  virtual void ac-
            tivemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller::tightMarshal2
            ( OpenWireFormat * wireFormat, commands::DataStructure
            * dataStructure, decaf::io::DataOutputStream * dataOut,
            utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

6.695 activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller Class Reference 3037

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1360).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller` (p. 1897), and `activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller` (p. 3216).

```
6.694.3.5 virtual void ac-
      tivemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller::tightUnmarshal
      ( OpenWireFormat * wireFormat, commands::DataStructure
      * dataStructure, decaf::io::DataInputStream * dataIn,
      utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1366).

Reimplemented in `activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller` (p. 1897), and `activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller` (p. 3216).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/TransactionIdMarshaller.h`

6.695 activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller Class Reference

Marshaling code for Open Wire Format for `TransactionIdMarshaller` (p. 3033).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/TransactionIdMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller`:

Public Member Functions

- `TransactionIdMarshaller ()`
- `virtual ~TransactionIdMarshaller ()`
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`

Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut, utils::BooleanStream *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.695.1 Detailed Description

Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3033). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.695.2 Constructor & Destructor Documentation

6.695.2.1 activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller::TransactionIdMarshaller () [inline]

6.695.2.2 virtual activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller::~~TransactionIdMarshaller () [inline, virtual]

6.695.3 Member Function Documentation

6.695.3.1 virtual void activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1342).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller** (p. 1884), and **activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller** (p. 3203).

6.695.3.2 virtual void activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1348).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller** (p. 1884), and **activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller** (p. 3203).

6.695.3.3 virtual int activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1354).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller` (p. 1884), and `activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller` (p. 3203).

```
6.695.3.4  virtual void ac-
            tivemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller::tightMarshal2
            ( OpenWireFormat * wireFormat, commands::DataStructure
              * dataStructure, decaf::io::DataOutputStream * dataOut,
              utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1360).

Reimplemented in `activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller` (p. 1885), and `activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller` (p. 3204).

```
6.695.3.5  virtual void ac-
            tivemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller::tightUnmarshal
            ( OpenWireFormat * wireFormat, commands::DataStructure
              * dataStructure, decaf::io::DataInputStream * dataIn,
              utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1366).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller** (p. 1885), and **activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller** (p. 3204).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/TransactionIdMarshaller.h`

6.696 activemq::commands::TransactionInfo Class Reference

```
#include <src/main/activemq/commands/TransactionInfo.h>
```

Inheritance diagram for **activemq::commands::TransactionInfo**:

Public Member Functions

- **TransactionInfo** ()
- virtual **~TransactionInfo** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaller share.
- virtual **TransactionInfo * cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this object's members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1372) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent.*
- virtual const **Pointer**< **ConnectionId** > & **getConnectionId** () const
- virtual **Pointer**< **ConnectionId** > & **getConnectionId** ()
- virtual void **setConnectionId** (const **Pointer**< **ConnectionId** > &connectionId)
- virtual const **Pointer**< **TransactionId** > & **getTransactionId** () const
- virtual **Pointer**< **TransactionId** > & **getTransactionId** ()
- virtual void **setTransactionId** (const **Pointer**< **TransactionId** > &transactionId)
- virtual unsigned char **getType** () const
- virtual void **setType** (unsigned char type)
- virtual bool **isTransactionInfo** () const
- virtual **Pointer**< **Command** > **visit** (activemq::state::CommandVisitor *visitor)
throw (exceptions::ActiveMQException)
Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_TRANSACTIONINFO** = 7

Protected Member Functions

- **TransactionInfo** (const **TransactionInfo** &)
- **TransactionInfo** & **operator=** (const **TransactionInfo** &)

Protected Attributes

- **Pointer**< **ConnectionId** > **connectionId**
- **Pointer**< **TransactionId** > **transactionId**
- unsigned char **type**

6.696.1 Constructor & Destructor Documentation

6.696.1.1 **activemq::commands::TransactionInfo::TransactionInfo** (const **TransactionInfo** &) [inline, protected]

6.696.1.2 **activemq::commands::TransactionInfo::TransactionInfo** ()

6.696.1.3 **virtual activemq::commands::TransactionInfo::~~TransactionInfo** () [virtual]

6.696.2 Member Function Documentation

6.696.2.1 **virtual TransactionInfo* activemq::commands::TransactionInfo::cloneDataStructure** () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1373).

6.696.2.2 **virtual void activemq::commands::TransactionInfo::copyDataStructure** (const **DataStructure** * *src*) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 598).

6.696.2.3 `virtual bool activemq::commands::TransactionInfo::equals (const DataStructure * value) const` [virtual]

Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 598).

6.696.2.4 `virtual Pointer<ConnectionId>& activemq::commands::TransactionInfo::getConnectionId ()` [virtual]

6.696.2.5 `virtual const Pointer<ConnectionId>& activemq::commands::TransactionInfo::getConnectionId () const` [virtual]

6.696.2.6 `virtual unsigned char activemq::commands::TransactionInfo::getDataStructureType () const` [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataStructure** (p. 1372) type copy.

Implements **activemq::commands::DataStructure** (p. 1375).

6.696.2.7 `virtual const Pointer<TransactionId>& activemq::commands::TransactionInfo::getTransactionId () const` [virtual]

6.696.2.8 `virtual Pointer<TransactionId>& activemq::commands::TransactionInfo::getTransactionId ()` [virtual]

6.696.2.9 `virtual unsigned char activemq::commands::TransactionInfo::getType () const` [virtual]

6.696.2.10 `virtual bool activemq::commands::TransactionInfo::isTransactionInfo () const` [inline, virtual]

Returns

an answer of true to the **isTransactionInfo()** (p. 3039) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 602).

- 6.696.2.11 `TransactionInfo& activemq::commands::TransactionInfo::operator= (const TransactionInfo &)` [inline, protected]
- 6.696.2.12 `virtual void activemq::commands::TransactionInfo::setConnectionId (const Pointer< ConnectionId > & connectionId)` [virtual]
- 6.696.2.13 `virtual void activemq::commands::TransactionInfo::setTransactionId (const Pointer< TransactionId > & transactionId)` [virtual]
- 6.696.2.14 `virtual void activemq::commands::TransactionInfo::setType (unsigned char type)` [virtual]
- 6.696.2.15 `virtual std::string activemq::commands::TransactionInfo::toString () const` [virtual]

Returns a string containing the information for this **DataStructure** (p.1372) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 602).

- 6.696.2.16 `virtual Pointer<Command> activemq::commands::TransactionInfo::visit (activemq::state::CommandVisitor * visitor) throw (exceptions::ActiveMQException)` [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper processXXX method in the visitor.

Returns

a **Response** (p. 2611) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p.995).

6.696.3 Field Documentation

- 6.696.3.1 `Pointer<ConnectionId> activemq::commands::TransactionInfo::connectionId` [protected]
- 6.696.3.2 `const unsigned char activemq::commands::TransactionInfo::ID_TRANSACTIONINFO = 7` [static]
- 6.696.3.3 `Pointer<TransactionId> activemq::commands::TransactionInfo::transactionId` [protected]
- 6.696.3.4 `unsigned char activemq::commands::TransactionInfo::type` [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/TransactionInfo.h

6.697 activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3041).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/TransactionInfoMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller:

Public Member Functions

- **TransactionInfoMarshaller** ()
- virtual **~TransactionInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.697.1 Detailed Description

Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3041). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.697.2 Constructor & Destructor Documentation

6.697.2.1 `activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller::TransactionInfoMarshaller () [inline]`

6.697.2.2 `virtual activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller::~~TransactionInfoMarshaller () [inline, virtual]`

6.697.3 Member Function Documentation

6.697.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.697.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.697.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 624).

6.697.3.4 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller::looseUnmarshal
(**OpenWireFormat** * *wireFormat*, **commands::DataStructure** *
dataStructure, **decaf::io::DataInputStream** * *dataIn*) throw (**decaf::io::IOException**) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 625).

6.697.3.5 virtual int ac-
tivemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller::tightMarshal1
(**OpenWireFormat** * *wireFormat*, **commands::DataStructure**
* *dataStructure*, **utils::BooleanStream** * *bs*) throw (**decaf::io::IOException**) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller** (p. 626).

```

6.697.3.6 virtual void ac-
    tivemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller::tightMarshal2
    ( OpenWireFormat * wireFormat, commands::DataStructure
    * dataStructure, decaf::io::DataOutputStream * dataOut,
    utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]

```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 628).

```

6.697.3.7 virtual void ac-
    tivemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller::tightUnmarshal
    ( OpenWireFormat * wireFormat, commands::DataStructure
    * dataStructure, decaf::io::DataInputStream * dataIn,
    utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]

```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller` (p. 629).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/TransactionInfoMarshaller.h`

6.698 activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller Class Reference

Marshaling code for Open Wire Format for `TransactionInfoMarshaller` (p. 3044).

#include <src/main/activemq/wireformat/openwire/marshal/v3/TransactionInfoMarshaller.h>

Inheritance diagram for activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller:

Public Member Functions

- **TransactionInfoMarshaller** ()
- virtual **~TransactionInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.698.1 Detailed Description

Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3044). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.698.2 Constructor & Destructor Documentation

6.698.2.1 `activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller::TransactionInfoMarshaller()` [inline]

6.698.2.2 `virtual activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller::~~TransactionInfoMarshaller()` [inline, virtual]

6.698.3 Member Function Documentation

6.698.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.698.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.698.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut)` throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 604).

6.698.3.4 virtual void activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 605).

6.698.3.5 virtual int activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 606).

6.698.3.6 virtual void activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 608).

```
6.698.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller` (p. 609).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/TransactionInfoMarshaller.h`

6.699 `activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller` Class Reference

Marshaling code for Open Wire Format for `TransactionInfoMarshaller` (p. 3048).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/TransactionInfoMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller`:

Public Member Functions

- **TransactionInfoMarshaller** ()
- virtual **~TransactionInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.699.1 Detailed Description

Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3048). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.699.2 Constructor & Destructor Documentation

6.699.2.1 `activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller::TransactionInfoMarshaller()` [inline]

6.699.2.2 `virtual activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller::~~TransactionInfoMarshaller()` [inline, virtual]

6.699.3 Member Function Documentation

6.699.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.699.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.699.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 611).

6.699.3.4 virtual void activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 612).

6.699.3.5 virtual int activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 613).

6.699.3.6 virtual void activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 614).

```
6.699.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller` (p. 615).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/TransactionInfoMarshaller.h`

6.700 `activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller` Class Reference

Marshaling code for Open Wire Format for `TransactionInfoMarshaller` (p. 3052).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/TransactionInfoMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller`:

Public Member Functions

- **TransactionInfoMarshaller** ()
- virtual **~TransactionInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.700.1 Detailed Description

Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3052). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.700.2 Constructor & Destructor Documentation

6.700.2.1 `activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller::TransactionInfoMarshaller()` [inline]

6.700.2.2 `virtual activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller::~~TransactionInfoMarshaller()` [inline, virtual]

6.700.3 Member Function Documentation

6.700.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.700.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.700.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 618).

6.700.3.4 virtual void activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 619).

6.700.3.5 virtual int activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller** (p. 620).

6.700.3.6 virtual void activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 621).

```
6.700.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller` (p. 622).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/TransactionInfoMarshaller.h`

6.701 `activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller` Class Reference

Marshaling code for Open Wire Format for `TransactionInfoMarshaller` (p. 3056).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/TransactionInfoMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller`:

Public Member Functions

- **TransactionInfoMarshaller** ()
- virtual **~TransactionInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**, **utils::BooleanStream *bs**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataInputStream *dataIn**) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat *wireFormat**, **commands::DataStructure *dataStructure**, **decaf::io::DataOutputStream *dataOut**) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.701.1 Detailed Description

Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3056). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.701.2 Constructor & Destructor Documentation

6.701.2.1 `activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller::TransactionInfoMarshaller()` [inline]

6.701.2.2 `virtual activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller::~~TransactionInfoMarshaller()` [inline, virtual]

6.701.3 Member Function Documentation

6.701.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller::createObject()` const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.701.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller::getDataStructureType()` const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.701.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 631).

6.701.3.4 virtual void activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 632).

6.701.3.5 virtual int activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 633).

6.701.3.6 virtual void activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

- wireFormat* - describes the wire format of the broker
- dataStructure* - Object to be marshaled
- dataOut* - BinaryReader that provides that data sink
- bs* - BooleanStream stream used to pack bits from the wire.

Exceptions

- IOException* if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 634).

```
6.701.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

- wireFormat* - describes the wire format of the broker.
- dataStructure* - Object to be un-marshaled.
- dataIn* - BinaryReader that provides that data.
- bs* - BooleanStream stream used to unpack bits from the wire.

Exceptions

- IOException* if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller` (p. 635).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/TransactionInfoMarshaller.h`

6.702 `activemq::state::TransactionState` Class Reference

```
#include <src/main/activemq/state/TransactionState.h>
```

Public Member Functions

- `TransactionState` (const `Pointer< TransactionId >` &id)
- virtual `~TransactionState` ()
- `std::string toString` () const
- void `addCommand` (const `Pointer< Command >` &operation)

- void **checkShutdown** () const
- void **shutdown** ()
- const **StlList**< **Pointer**< **Command** > > & **getCommands** () const
- const **Pointer**< **TransactionId** > & **getId** () const
- void **setPrepared** (bool prepared)
- bool **isPrepared** () const
- void **setPreparedResult** (int preparedResult)
- int **getPreparedResult** () const

6.702.1 Constructor & Destructor Documentation

6.702.1.1 **activemq::state::TransactionState::TransactionState** (const **Pointer**< **TransactionId** > & *id*)

6.702.1.2 **virtual activemq::state::TransactionState::~~TransactionState** ()
[virtual]

6.702.2 Member Function Documentation

6.702.2.1 **void activemq::state::TransactionState::addCommand** (const **Pointer**< **Command** > & *operation*)

6.702.2.2 **void activemq::state::TransactionState::checkShutdown** () const

6.702.2.3 **const StlList**< **Pointer**<**Command**> >& **activemq::state::TransactionState::getCommands** ()
const [inline]

6.702.2.4 **const Pointer**<**TransactionId**>& **activemq::state::TransactionState::getId** () const [inline]

6.702.2.5 **int activemq::state::TransactionState::getPreparedResult** () const
[inline]

6.702.2.6 **bool activemq::state::TransactionState::isPrepared** () const [inline]

6.702.2.7 **void activemq::state::TransactionState::setPrepared** (bool *prepared*)
[inline]

6.702.2.8 **void activemq::state::TransactionState::setPreparedResult** (int *preparedResult*) [inline]

6.702.2.9 **void activemq::state::TransactionState::shutdown** () [inline]

6.702.2.10 **std::string activemq::state::TransactionState::toString** () const

The documentation for this class was generated from the following file:

- src/main/activemq/state/**TransactionState.h**

6.703 decaf::internal::util::concurrent::Transferer< E > Class Template Reference

Shared internal API for dual stacks and queues.

```
#include <src/main/decaf/internal/util/concurrent/Transferer.h>
```

Inheritance diagram for decaf::internal::util::concurrent::Transferer< E >:

6.703.1 Detailed Description

```
template<typename E> class decaf::internal::util::concurrent::Transferer< E >
```

Shared internal API for dual stacks and queues.

The documentation for this class was generated from the following file:

- src/main/decaf/internal/util/concurrent/**Transferer.h**

6.704 decaf::internal::util::concurrent::TransferQueue< E > Class Template Reference

This extends Scherer-Scott dual queue algorithm, differing, among other ways, by using modes within nodes rather than marked pointers.

```
#include <src/main/decaf/internal/util/concurrent/TransferQueue.h>
```

Inheritance diagram for decaf::internal::util::concurrent::TransferQueue< E >:

Public Member Functions

- **TransferQueue** ()
*Node class for **TransferQueue** (p. 3062).*
- virtual ~**TransferQueue** ()
- virtual void **transfer** (E *e, bool timed, long long nanos) throw (decaf::util::concurrent::TimeoutException, decaf::lang::exceptions::InterruptedException)
Performs a put.
- virtual E * **transfer** (bool timed, long long nanos) throw (decaf::util::concurrent::TimeoutException, decaf::lang::exceptions::InterruptedException)
Performs a take.

6.704.1 Detailed Description

template<typename E> class decaf::internal::util::concurrent::TransferQueue< E >

This extends Scherer-Scott dual queue algorithm, differing, among other ways, by using nodes within nodes rather than marked pointers. The algorithm is a little simpler than that for stacks because fulfillers do not need explicit nodes, and matching is done by CAS'ing QNode.item field from non-null to null (for put) or vice versa (for take).

6.704.2 Constructor & Destructor Documentation

6.704.2.1 template<typename E > decaf::internal::util::concurrent::TransferQueue< E >::TransferQueue () [inline]

Node class for **TransferQueue** (p. 3062).

Tries to cancel by CAS'ing ref to NULL if that succeeds then we mark as cancelled. Returns true if this node is known to be off the queue because its next pointer has been forgotten due to an advanceHead operation.

6.704.2.2 template<typename E > virtual decaf::internal::util::concurrent::TransferQueue< E >::~~TransferQueue () [inline, virtual]

6.704.3 Member Function Documentation

6.704.3.1 template<typename E > virtual void decaf::internal::util::concurrent::TransferQueue< E >::transfer (E * *e*, bool *timed*, long long *nanos*) throw (decaf::util::concurrent::TimeoutException, decaf::lang::exceptions::InterruptedException) [inline, virtual]

Performs a put.

Parameters

e the item to be handed to a consumer;

timed if this operation should timeout

nanos the timeout, in nanoseconds

Exceptions

TimeoutException if the operation timed out waiting for the consumer to accept the item offered.

InterruptedException if the thread was interrupted while waiting for the consumer to accept the item offered.

Implements **decaf::internal::util::concurrent::Transferer< E >** (p. 3062).

```

6.704.3.2  template<typename E > virtual E*
             decaf::internal::util::concurrent::TransferQueue< E
             >::transfer ( bool timed, long long nanos )
             throw ( decaf::util::concurrent::TimeoutException,
             decaf::lang::exceptions::InterruptedException ) [inline, virtual]

```

Performs a take.

Parameters

timed if this operation should timeout
nanos the timeout, in nanoseconds

Returns

the item provided or received;

Exceptions

TimeoutException if the operation timed out waiting for the producer to offer an item.
InterruptedException if the thread was interrupted while waiting for the producer to offer an item.

Implements **decaf::internal::util::concurrent::Transferer< E >** (p. 3062).

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/util/concurrent/TransferQueue.h`

6.705 decaf::internal::util::concurrent::TransferStack< E > Class Template Reference

```
#include <src/main/decaf/internal/util/concurrent/TransferStack.h>
```

Inheritance diagram for `decaf::internal::util::concurrent::TransferStack< E >`:

Public Member Functions

- **TransferStack** ()
- virtual **~TransferStack** ()
- virtual void **transfer** (E *e, bool *timed*, long long *nanos*) throw (decaf::util::concurrent::TimeoutException, decaf::lang::exceptions::InterruptedException)
Performs a put.
- virtual E * **transfer** (bool *timed*, long long *nanos*) throw (decaf::util::concurrent::TimeoutException, decaf::lang::exceptions::InterruptedException)
Performs a take.

```
template<typename E> class decaf::internal::util::concurrent::TransferStack< E >
```

6.705.1 Constructor & Destructor Documentation

6.705.1.1 `template<typename E > decaf::internal::util::concurrent::TransferStack< E >::TransferStack () [inline]`

6.705.1.2 `template<typename E > virtual decaf::internal::util::concurrent::TransferStack< E >::~~TransferStack () [inline, virtual]`

6.705.2 Member Function Documentation

6.705.2.1 `template<typename E > virtual void decaf::internal::util::concurrent::TransferStack< E >::transfer (E * e, bool timed, long long nanos) throw (decaf::util::concurrent::TimeoutException, decaf::lang::exceptions::InterruptedException) [inline, virtual]`

Performs a put.

Parameters

e the item to be handed to a consumer;
timed if this operation should timeout
nanos the timeout, in nanoseconds

Exceptions

TimeoutException if the operation timed out waiting for the consumer to accept the item offered.
InterruptedException if the thread was interrupted while waiting for the consumer to accept the item offered.

Implements `decaf::internal::util::concurrent::Transferer< E >` (p. 3062).

6.705.2.2 `template<typename E > virtual E* decaf::internal::util::concurrent::TransferStack< E >::transfer (bool timed, long long nanos) throw (decaf::util::concurrent::TimeoutException, decaf::lang::exceptions::InterruptedException) [inline, virtual]`

Performs a take.

Parameters

timed if this operation should timeout
nanos the timeout, in nanoseconds

Returns

the item provided or received;

Exceptions

TimeoutException if the operation timed out waiting for the producer to offer an item.

InterruptedException if the thread was interrupted while waiting for the producer to offer an item.

Implements **decaf::internal::util::concurrent::Transferer< E >** (p. 3062).

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/util/concurrent/TransferStack.h`

6.706 activemq::transport::Transport Class Reference

Interface for a transport layer for command objects.

`#include <src/main/activemq/transport/Transport.h>`

Inheritance diagram for `activemq::transport::Transport`:

Public Member Functions

- virtual `~Transport()`
- virtual void **start** ()=0 throw (`decaf::io::IOException`)
*Starts the **Transport** (p. 3066), the send methods of a **Transport** (p. 3066) will throw an exception if used before the **Transport** (p. 3066) is started.*
- virtual void **stop** ()=0 throw (`decaf::io::IOException`)
*Stops the **Transport** (p. 3066).*
- virtual void **oneway** (const **Pointer**< **Command** > &command)=0 throw (`decaf::io::IOException`, `decaf::lang::exceptions::UnsupportedOperationException`)
Sends a one-way command.
- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command)=0 throw (`decaf::io::IOException`, `decaf::lang::exceptions::UnsupportedOperationException`)
Sends the given command to the broker and then waits for the response.
- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command, unsigned int timeout)=0 throw (`decaf::io::IOException`, `decaf::lang::exceptions::UnsupportedOperationException`)
Sends the given command to the broker and then waits for the response.
- virtual void **setWireFormat** (const **Pointer**< **wireformat::WireFormat** > &wireFormat)=0
Sets the WireFormat instance to use.
- virtual void **setTransportListener** (**TransportListener** *listener)=0
Sets the observer of asynchronous events from this transport.

- virtual **TransportListener** * **getTransportListener** () const =0
Gets the observer of asynchronous events from this transport.
- virtual **Transport** * **narrow** (const std::type_info &typeId)=0
*Narrows down a Chain of Transports to a specific **Transport** (p. 3066) to allow a higher level transport to skip intermediate Transports in certain circumstances.*
- virtual bool **isFaultTolerant** () const =0
*Is this **Transport** (p. 3066) fault tolerant, meaning that it will reconnect to a broker on disconnect.*
- virtual bool **isConnected** () const =0
*Is the **Transport** (p. 3066) Connected to its Broker.*
- virtual bool **isClosed** () const =0
*Has the **Transport** (p. 3066) been shutdown and no longer usable.*
- virtual std::string **getRemoteAddress** () const =0
- virtual void **reconnect** (const **decaf::net::URI** &uri)=0 throw (**decaf::io::IOException**)
reconnect to another location

6.706.1 Detailed Description

Interface for a transport layer for command objects. Callers can send oneway messages or make synchronous requests. Non-response messages will be delivered to the specified listener object upon receipt. A user of the **Transport** (p. 3066) can set an exception listener to be notified of errors that occurs in Threads that the **Transport** (p. 3066) layer runs. Transports should be given an instance of a WireFormat object when created so that they can turn the built in Commands to / from the required wire format encoding.

6.706.2 Constructor & Destructor Documentation

- 6.706.2.1** virtual **activemq::transport::Transport::~~Transport** () [inline, virtual]

6.706.3 Member Function Documentation

- 6.706.3.1** virtual std::string **activemq::transport::Transport::getRemoteAddress** () const [pure virtual]

Returns

the remote address for this connection

Implemented in **activemq::transport::mock::MockTransport** (p. 2231).

6.706.3.2 `virtual TransportListener* activemq::transport::Transport::getTransportListener () const [pure virtual]`

Gets the observer of asynchronous events from this transport.

Returns

the listener of transport events.

Implemented in `activemq::transport::mock::MockTransport` (p. 2231).

6.706.3.3 `virtual bool activemq::transport::Transport::isClosed () const [pure virtual]`

Has the **Transport** (p. 3066) been shutdown and no longer usable.

Returns

true if the **Transport** (p. 3066)

Implemented in `activemq::transport::mock::MockTransport` (p. 2232).

6.706.3.4 `virtual bool activemq::transport::Transport::isConnected () const [pure virtual]`

Is the **Transport** (p. 3066) Connected to its Broker.

Returns

true if a connection has been made.

Implemented in `activemq::transport::mock::MockTransport` (p. 2232).

6.706.3.5 `virtual bool activemq::transport::Transport::isFaultTolerant () const [pure virtual]`

Is this **Transport** (p. 3066) fault tolerant, meaning that it will reconnect to a broker on disconnect.

Returns

true if the **Transport** (p. 3066) is fault tolerant.

Implemented in `activemq::transport::mock::MockTransport` (p. 2232).

6.706.3.6 `virtual Transport* activemq::transport::Transport::narrow (const std::type_info & typeId) [pure virtual]`

Narrows down a Chain of Transports to a specific **Transport** (p. 3066) to allow a higher level transport to skip intermediate Transports in certain circumstances.

Parameters

typeId - The type_info of the Object we are searching for.

Returns

the requested Object. or NULL if its not in this chain.

Implemented in **activemq::transport::mock::MockTransport** (p. 2233).

Referenced by **activemq::transport::failover::FailoverTransport::narrow()**.

6.706.3.7 `virtual void activemq::transport::Transport::oneway (const Pointer< Command > & command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [pure virtual]`

Sends a one-way command.

Does not wait for any response from the broker.

Parameters

command the command to be sent.

Exceptions

IOException if an exception occurs during writing of the command.

UnsupportedOperationException if this method is not implemented by this transport.

Implemented in **activemq::transport::mock::MockTransport** (p. 2233).

6.706.3.8 `virtual void activemq::transport::Transport::reconnect (const decaf::net::URI & uri) throw (decaf::io::IOException) [pure virtual]`

reconnect to another location

Parameters

uri

Exceptions

IOException on failure of if not supported

6.706.3.9 `virtual Pointer<Response> activemq::transport::Transport::request (const Pointer< Command > & command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [pure virtual]`

Sends the given command to the broker and then waits for the response.

Parameters

command the command to be sent.

Returns

the response from the broker.

Exceptions

IOException if an exception occurs during the read of the command.

UnsupportedOperationException if this method is not implemented by this transport.

Implemented in **activemq::transport::mock::MockTransport** (p. 2234).

```
6.706.3.10  virtual Pointer<Response> activemq::transport::Transport::request  
              ( const Pointer< Command > & command, unsigned  
                int timeout ) throw ( decaf::io::IOException,  
                decaf::lang::exceptions::UnsupportedOperationException ) [pure  
                virtual]
```

Sends the given command to the broker and then waits for the response.

Parameters

command - The command to be sent.

timeout - The time to wait for this response.

Returns

the response from the broker.

Exceptions

IOException if an exception occurs during the read of the command.

UnsupportedOperationException if this method is not implemented by this transport.

Implemented in **activemq::transport::mock::MockTransport** (p. 2234).

```
6.706.3.11  virtual void activemq::transport::Transport::setTransportListener (   
              TransportListener * listener ) [pure virtual]
```

Sets the observer of asynchronous events from this transport.

Parameters

listener the listener of transport events.

Implemented in **activemq::transport::mock::MockTransport** (p. 2236).

```
6.706.3.12  virtual void activemq::transport::Transport::setWireFormat ( const  
              Pointer< wireformat::WireFormat > & wireFormat ) [pure virtual]
```

Sets the WireFormat instance to use.

Parameters

wireFormat The WireFormat the object used to encode / decode commands.

6.706.3.13 virtual void activemq::transport::Transport::start () throw (decaf::io::IOException) [pure virtual]

Starts the **Transport** (p. 3066), the send methods of a **Transport** (p. 3066) will throw an exception if used before the **Transport** (p. 3066) is started.

Exceptions

IOException if and error occurs while starting the **Transport** (p. 3066).

Implemented in **activemq::transport::mock::MockTransport** (p. 2236).

6.706.3.14 virtual void activemq::transport::Transport::stop () throw (decaf::io::IOException) [pure virtual]

Stops the **Transport** (p. 3066).

Exceptions

IOException if an error occurs while stopping the transport.

Implemented in **activemq::transport::mock::MockTransport** (p. 2236).

The documentation for this class was generated from the following file:

- src/main/activemq/transport/**Transport.h**

6.707 activemq::transport::TransportFactory Class Reference

Defines the interface for Factories that create Transports or TransportFilters.

```
#include <src/main/activemq/transport/TransportFactory.h>
```

Inheritance diagram for activemq::transport::TransportFactory:

Public Member Functions

- virtual ~**TransportFactory** ()
- virtual **Pointer**< **Transport** > **create** (const **decaf::net::URI** &location)=0 throw (exceptions::ActiveMQException)
*Creates a fully configured **Transport** (p. 3066) instance which could be a chain of filters and transports.*
- virtual **Pointer**< **Transport** > **createComposite** (const **decaf::net::URI** &location)=0 throw (exceptions::ActiveMQException)
*Creates a slimed down **Transport** (p. 3066) instance which can be used in composite transport instances.*

6.707.1 Detailed Description

Defines the interface for Factories that create **Transports** or **TransportFilters**. The factory should be able to create either a completely configured **Transport** (p. 3066) meaning that it has all the appropriate filters wrapping it, or it should be able to create a slimed down version that is used in composite transports like Failover or Fanout.

Since

3.0

6.707.2 Constructor & Destructor Documentation

6.707.2.1 `virtual activemq::transport::TransportFactory::~~TransportFactory ()`
[inline, virtual]

6.707.3 Member Function Documentation

6.707.3.1 `virtual Pointer<Transport> activemq::transport::TransportFactory::create (const decaf::net::URI & location) throw (exceptions::ActiveMQException)`
[pure virtual]

Creates a fully configured **Transport** (p. 3066) instance which could be a chain of filters and transports.

Parameters

location - URI location to connect to plus any properties to assign.

Exceptions

ActiveMQException if an error occurs

Implemented in `activemq::transport::failover::FailoverTransportFactory` (p. 1524), `activemq::transport::mock::MockTransportFactory` (p. 2238), and `activemq::transport::tcp::TcpTransportFactory` (p. 2970).

6.707.3.2 `virtual Pointer<Transport> activemq::transport::TransportFactory::createComposite (const decaf::net::URI & location) throw (exceptions::ActiveMQException)` [pure virtual]

Creates a slimed down **Transport** (p. 3066) instance which can be used in composite transport instances.

Parameters

location - URI location to connect to plus any properties to assign.

Exceptions

ActiveMQException if an error occurs

Implemented in **activemq::transport::failover::FailoverTransportFactory** (p. 1524), **activemq::transport::mock::MockTransportFactory** (p. 2238), and **activemq::transport::tcp::TcpTransportFactory** (p. 2970).

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/TransportFactory.h`

6.708 activemq::transport::TransportFilter Class Reference

A filter on the transport layer.

```
#include <src/main/activemq/transport/TransportFilter.h>
```

Inheritance diagram for `activemq::transport::TransportFilter`:

Public Member Functions

- **TransportFilter** (const **Pointer**< **Transport** > &next)
Constructor.
- virtual **~TransportFilter** ()
- virtual void **onCommand** (const **Pointer**< **Command** > &command)
Event handler for the receipt of a command.
- virtual void **onException** (const **decaf::lang::Exception** &ex)
Event handler for an exception from a command transport.
- virtual void **transportInterrupted** ()
The transport has suffered an interruption from which it hopes to recover.
- virtual void **transportResumed** ()
The transport has resumed after an interruption.
- virtual void **oneway** (const **Pointer**< **Command** > &command) throw (**decaf::io::IOException**, **decaf::lang::exceptions::UnsupportedOperationException**)
Sends a one-way command.
- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command) throw (**decaf::io::IOException**, **decaf::lang::exceptions::UnsupportedOperationException**)
Not supported by this class - throws an exception.
- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command, unsigned int timeout) throw (**decaf::io::IOException**, **decaf::lang::exceptions::UnsupportedOperationException**)
Not supported by this class - throws an exception.
- virtual void **setTransportListener** (**TransportListener** *listener)
Sets the observer of asynchronous exceptions from this transport.

- virtual **TransportListener** * **getTransportListener** () const
Gets the observer of asynchronous exceptions from this transport.
- virtual void **setWireFormat** (const **Pointer**< **wireformat::WireFormat** > &wireFormat)
Sets the WireFormat instance to use.
- virtual void **start** () throw (decaf::io::IOException)
Starts this transport object and creates the thread for polling on the input stream for commands.
- virtual void **stop** () throw (decaf::io::IOException)
*Stops the **Transport** (p. 3066).*
- virtual void **close** () throw (decaf::io::IOException)
Stops the polling thread and closes the streams.
- virtual **Transport** * **narrow** (const std::type_info &typeId)
*Narrows down a Chain of Transports to a specific **Transport** (p. 3066) to allow a higher level transport to skip intermediate Transports in certain circumstances.*
- virtual bool **isFaultTolerant** () const
*Is this **Transport** (p. 3066) fault tolerant, meaning that it will reconnect to a broker on disconnect.*
- virtual bool **isConnected** () const
*Is the **Transport** (p. 3066) Connected to its Broker.*
- virtual bool **isClosed** () const
*Has the **Transport** (p. 3066) been shutdown and no longer usable.*
- virtual std::string **getRemoteAddress** () const
- virtual void **reconnect** (const **decaf::net::URI** &uri) throw (decaf::io::IOException)
reconnect to another location

Protected Member Functions

- void **fire** (const **decaf::lang::Exception** &ex)
Notify the listener of the thrown Exception.
- void **fire** (const **Pointer**< **Command** > &command)
Notify the listener of the new incoming Command.

Protected Attributes

- **Pointer**< **Transport** > **next**
The transport that this filter wraps around.

- **TransportListener * listener**

Listener of this transport.

6.708.1 Detailed Description

A filter on the transport layer. **Transport** (p.3066) filters implement the **Transport** (p.3066) interface and optionally delegate calls to another **Transport** (p.3066) object.

Since

1.0

6.708.2 Constructor & Destructor Documentation

6.708.2.1 **activemq::transport::TransportFilter::TransportFilter** (**const Pointer< Transport > & next**)

Constructor.

Parameters

next - the next **Transport** (p.3066) in the chain

6.708.2.2 **virtual activemq::transport::TransportFilter::~~TransportFilter** () [inline, virtual]

6.708.3 Member Function Documentation

6.708.3.1 **virtual void activemq::transport::TransportFilter::close** () **throw** (**decaf::io::IOException**) [virtual]

Stops the polling thread and closes the streams.

This can be called explicitly, but is also called in the destructor. Once this object has been closed, it cannot be restarted.

Exceptions

IOException if an error occurs while closing the **Transport** (p.3066).

Reimplemented in **activemq::transport::correlator::ResponseCorrelator** (p.2617), **activemq::transport::inactivity::InactivityMonitor** (p.1625), **activemq::transport::tcp::TcpTransport** (p.2968), and **activemq::wireformat::openwire::OpenWireFormatNegotiator** (p.2311).

6.708.3.2 **void activemq::transport::TransportFilter::fire** (**const Pointer< Command > & command**) [protected]

Notify the listener of the new incoming Command.

Parameters

command - the command to send to the listener

6.708.3.3 `void activemq::transport::TransportFilter::fire (const decaf::lang::Exception & ex) [protected]`

Notify the listener of the thrown Exception.

Parameters

ex - the exception to send to listeners

6.708.3.4 `virtual std::string activemq::transport::TransportFilter::getRemoteAddress () const [inline, virtual]`

Returns

the remote address for this connection

6.708.3.5 `virtual TransportListener* activemq::transport::TransportFilter::getTransportListener () const [inline, virtual]`

Gets the observer of asynchronous exceptions from this transport.

Returns

The listener of transport events.

6.708.3.6 `virtual bool activemq::transport::TransportFilter::isClosed () const [inline, virtual]`

Has the **Transport** (p. 3066) been shutdown and no longer usable.

Returns

true if the **Transport** (p. 3066)

Reimplemented in **activemq::transport::tcp::TcpTransport** (p. 2968).

6.708.3.7 `virtual bool activemq::transport::TransportFilter::isConnected () const [inline, virtual]`

Is the **Transport** (p. 3066) Connected to its Broker.

Returns

true if a connection has been made.

Reimplemented in **activemq::transport::tcp::TcpTransport** (p. 2969).

6.708.3.8 `virtual bool activemq::transport::TransportFilter::isFaultTolerant ()
const [inline, virtual]`

Is this **Transport** (p. 3066) fault tolerant, meaning that it will reconnect to a broker on disconnect.

Returns

true if the **Transport** (p. 3066) is fault tolerant.

Reimplemented in **activemq::transport::tcp::TcpTransport** (p. 2969).

6.708.3.9 `virtual Transport* activemq::transport::TransportFilter::narrow (const
std::type_info & typeId) [virtual]`

Narrows down a Chain of Transports to a specific **Transport** (p. 3066) to allow a higher level transport to skip intermediate Transports in certain circumstances.

Parameters

typeId - The type_info of the Object we are searching for.

Returns

the requested Object. or NULL if its not in this chain.

6.708.3.10 `virtual void activemq::transport::TransportFilter::onCommand (const
Pointer< Command > & command) [virtual]`

Event handler for the receipt of a command.

Parameters

command - the received command object.

Implements **activemq::transport::TransportListener** (p. 3081).

Reimplemented in **activemq::transport::correlator::ResponseCorrelator** (p. 2618), **activemq::transport::inactivity::InactivityMonitor** (p. 1626), **activemq::transport::logging::LoggingTransport** (p. 1921), and **activemq::wireformat::openwire::OpenWireFormatNegotiator** (p. 2311).

6.708.3.11 `virtual void activemq::transport::TransportFilter::oneway (const
Pointer< Command > & command) throw (decaf::io::IOException,
decaf::lang::exceptions::UnsupportedOperationException) [inline,
virtual]`

Sends a one-way command.

Does not wait for any response from the broker.

Parameters

command the command to be sent.

Exceptions

IOException if an exception occurs during writing of the command.

UnsupportedOperationException if this method is not implemented by this transport.

Reimplemented in **activemq::transport::correlator::ResponseCorrelator** (p. 2618), **activemq::transport::inactivity::InactivityMonitor** (p. 1626), **activemq::transport::logging::LoggingTransport** (p. 1922), and **activemq::wireformat::openwire::OpenWireFormatNegotiator** (p. 2311).

6.708.3.12 `virtual void activemq::transport::TransportFilter::onException (const decaf::lang::Exception & ex) [virtual]`

Event handler for an exception from a command transport.

Parameters

ex The exception to handle.

Implements **activemq::transport::TransportListener** (p. 3082).

Reimplemented in **activemq::transport::inactivity::InactivityMonitor** (p. 1626).

6.708.3.13 `virtual void activemq::transport::TransportFilter::reconnect (const decaf::net::URI & uri) throw (decaf::io::IOException) [virtual]`

reconnect to another location

Parameters

uri

Exceptions

IOException on failure of if not supported

6.708.3.14 `virtual Pointer<Response> activemq::transport::TransportFilter::request (const Pointer< Command > & command, unsigned int timeout) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [inline, virtual]`

Not supported by this class - throws an exception.

Parameters

command - The command that is sent as a request

timeout - The the time to wait for a response.

Exceptions

IOException

UnsupportedOperationException.

Reimplemented in `activemq::transport::correlator::ResponseCorrelator` (p. 2619), `activemq::transport::logging::LoggingTransport` (p. 1922), and `activemq::wireformat::openwire::OpenWireFormatNegotiator` (p. 2312).

6.708.3.15 `virtual Pointer<Response> activemq::transport::TransportFilter::request (const Pointer< Command > & command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [inline, virtual]`

Not supported by this class - throws an exception.

Parameters

command the command that is sent as a request

Exceptions

IOException

UnsupportedOperationException.

Reimplemented in `activemq::transport::correlator::ResponseCorrelator` (p. 2618), `activemq::transport::logging::LoggingTransport` (p. 1922), and `activemq::wireformat::openwire::OpenWireFormatNegotiator` (p. 2312).

6.708.3.16 `virtual void activemq::transport::TransportFilter::setTransportListener (TransportListener * listener) [inline, virtual]`

Sets the observer of asynchronous exceptions from this transport.

Parameters

listener the listener of transport events.

6.708.3.17 `virtual void activemq::transport::TransportFilter::setWireFormat (const Pointer< wireformat::WireFormat > & wireFormat) [inline, virtual]`

Sets the WireFormat instance to use.

Parameters

wireFormat The WireFormat the object used to encode / decode commands.

6.708.3.18 `virtual void activemq::transport::TransportFilter::start () throw (decaf::io::IOException) [virtual]`

Starts this transport object and creates the thread for polling on the input stream for commands.

If this object has been closed, throws an exception. Before calling start, the caller must set the IO streams and the reader and writer objects.

Exceptions

IOException if an error occurs or if this transport has already been closed.

Reimplemented in **activemq::transport::correlator::ResponseCorrelator** (p. 2619), and **activemq::wireformat::openwire::OpenWireFormatNegotiator** (p. 2313).

6.708.3.19 `virtual void activemq::transport::TransportFilter::stop () throw (decaf::io::IOException) [virtual]`

Stops the **Transport** (p. 3066).

Exceptions

IOException if an error occurs while stopping the **Transport** (p. 3066).

6.708.3.20 `virtual void activemq::transport::TransportFilter::transportInterrupted () [virtual]`

The transport has suffered an interruption from which it hopes to recover.

Implements **activemq::transport::TransportListener** (p. 3082).

6.708.3.21 `virtual void activemq::transport::TransportFilter::transportResumed () [virtual]`

The transport has resumed after an interruption.

Implements **activemq::transport::TransportListener** (p. 3082).

6.708.4 Field Documentation

6.708.4.1 `TransportListener* activemq::transport::TransportFilter::listener [protected]`

Listener of this transport.

6.708.4.2 `Pointer<Transport> activemq::transport::TransportFilter::next [protected]`

The transport that this filter wraps around.

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/TransportFilter.h`

6.709 activemq::transport::TransportListener Class Reference

A listener of asynchronous exceptions from a command transport object.

```
#include <src/main/activemq/transport/TransportListener.h>
```

Inheritance diagram for activemq::transport::TransportListener:

Public Member Functions

- virtual `~TransportListener ()`
- virtual void `onCommand (const Pointer< Command > &command)=0`
Event handler for the receipt of a command.
- virtual void `onException (const decaf::lang::Exception &ex)=0`
Event handler for an exception from a command transport.
- virtual void `transportInterrupted ()=0`
The transport has suffered an interruption from which it hopes to recover.
- virtual void `transportResumed ()=0`
The transport has resumed after an interruption.

6.709.1 Detailed Description

A listener of asynchronous exceptions from a command transport object.

6.709.2 Constructor & Destructor Documentation

- 6.709.2.1** virtual `activemq::transport::TransportListener::~~TransportListener ()`
[inline, virtual]

6.709.3 Member Function Documentation

- 6.709.3.1** virtual void `activemq::transport::TransportListener::onCommand (const Pointer< Command > & command)` [pure virtual]

Event handler for the receipt of a command.

The transport passes off all received commands to its listeners, the listener then owns the Object. If there is no registered listener the **Transport** (p. 3066) deletes the command upon receipt.

Parameters

command the received command object.

Implemented in `activemq::core::ActiveMQConnection` (p. 209), `activemq::transport::correlator::ResponseCorrelator` (p. 2618), `activemq::transport::failover::FailoverTransportListener` (p. 1526), `activemq::transport::inactivity::InactivityMonitor` (p. 1626), `activemq::transport::logging::LoggingTransport` (p. 1921), `activemq::transport::mock::InternalCommandListener` (p. 1690), `activemq::transport::TransportFilter` (p. 3077), and `activemq::wireformat::openwire::OpenWireFormatNegotiator` (p. 2311).

6.709.3.2 virtual void activemq::transport::TransportListener::onException (const decaf::lang::Exception & *ex*) [pure virtual]

Event handler for an exception from a command transport.

Parameters

ex The exception being propagated to this listener to handle.

Implemented in **activemq::core::ActiveMQConnection** (p. 209),
activemq::transport::failover::BackupTransport (p. 593), **ac-**
tivemq::transport::failover::FailoverTransportListener (p. 1526), **ac-**
tivemq::transport::inactivity::InactivityMonitor (p. 1626), and **ac-**
tivemq::transport::TransportFilter (p. 3078).

6.709.3.3 virtual void activemq::transport::TransportListener::transportInterrupted () [pure virtual]

The transport has suffered an interruption from which it hopes to recover.

Implemented in **activemq::core::ActiveMQConnection** (p. 211),
activemq::transport::DefaultTransportListener (p. 1385), **ac-**
tivemq::transport::failover::FailoverTransportListener (p. 1526), and **ac-**
tivemq::transport::TransportFilter (p. 3080).

6.709.3.4 virtual void activemq::transport::TransportListener::transportResumed () [pure virtual]

The transport has resumed after an interruption.

Implemented in **activemq::core::ActiveMQConnection** (p. 211),
activemq::transport::DefaultTransportListener (p. 1385), **ac-**
tivemq::transport::failover::FailoverTransportListener (p. 1526), and **ac-**
tivemq::transport::TransportFilter (p. 3080).

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/TransportListener.h`

6.710 activemq::transport::TransportRegistry Class Reference

Registry of all **Transport** (p. 3066) Factories that are available to the client at runtime.

```
#include <src/main/activemq/transport/TransportRegistry.h>
```

Public Member Functions

- virtual `~TransportRegistry ()`
- `TransportFactory * findFactory (const std::string &name) const throw (decaf::lang::exceptions::NoSuchElementException)`

*Gets a Registered **TransportFactory** (p. 3071) from the Registry and returns it if there is not a registered format factory with the given name an exception is thrown.*

- void **registerFactory** (const std::string &name, **TransportFactory** *factory) throw (decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::NullPointerException)

*Registers a new **TransportFactory** (p. 3071) with this Registry.*

- void **unregisterFactory** (const std::string &name)

Unregisters the Factory with the given name and deletes that instance of the Factory.

- std::vector< std::string > **getTransportNames** () const

Retrieves a list of the names of all the Registered Transport's in this Registry.

Static Public Member Functions

- static **TransportRegistry & getInstance** ()

*Gets the single instance of the **TransportRegistry** (p. 3082).*

6.710.1 Detailed Description

Registry of all **Transport** (p. 3066) Factories that are available to the client at runtime. New Transport's must have a factory registered here before a connection attempt is made.

Since

3.0

6.710.2 Constructor & Destructor Documentation

- 6.710.2.1 virtual **activemq::transport::TransportRegistry::~TransportRegistry** ()
[virtual]

6.710.3 Member Function Documentation

- 6.710.3.1 **TransportFactory* activemq::transport::TransportRegistry::findFactory**
(const std::string & *name*) const throw (decaf::lang::exceptions::NoSuchElementException)

Gets a Registered **TransportFactory** (p. 3071) from the Registry and returns it if there is not a registered format factory with the given name an exception is thrown.

Parameters

name The name of the Factory to find in the Registry.

Returns

the Factory registered under the given name.

Exceptions

NoSuchElementException if no factory is registered with that name.

6.710.3.2 `static TransportRegistry& activemq::transport::TransportRegistry::getInstance ()`
[static]

Gets the single instance of the **TransportRegistry** (p. 3082).

Returns

reference to the single instance of this Registry

6.710.3.3 `std::vector<std::string> activemq::transport::TransportRegistry::getTransportNames () const`

Retrieves a list of the names of all the Registered Transport's in this Registry.

Returns

stl vector of strings with all the **Transport** (p. 3066) names registered.

6.710.3.4 `void activemq::transport::TransportRegistry::registerFactory (const std::string & name, TransportFactory * factory) throw (decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::NullPointerException)`

Registers a new **TransportFactory** (p. 3071) with this Registry.

If a Factory with the given name is already registered it is overwritten with the new one. Once a factory is added to the Registry its lifetime is controlled by the Registry, it will be deleted once the Registry has been deleted.

Parameters

name The name of the new Factory to register.

factory The new Factory to add to the Registry.

Exceptions

IllegalArgumentException is name is the empty string.

NullPointerException if the Factory is Null.

6.710.3.5 `void activemq::transport::TransportRegistry::unregisterFactory (const std::string & name)`

Unregisters the Factory with the given name and deletes that instance of the Factory.

Parameters

- name* Name of the Factory to unregister and destroy

The documentation for this class was generated from the following file:

- src/main/activemq/transport/**TransportRegistry.h**

6.711 decaf::net::UnknownHostException Class Reference

```
#include <src/main/decaf/net/UnknownHostException.h>
```

Inheritance diagram for decaf::net::UnknownHostException:

Public Member Functions

- **UnknownHostException** () throw ()
Default Constructor.
- **UnknownHostException** (const Exception &ex) throw ()
Conversion Constructor from some other Exception.
- **UnknownHostException** (const **UnknownHostException** &ex) throw ()
Copy Constructor.
- **UnknownHostException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **UnknownHostException** (const std::exception *cause) throw ()
Constructor.
- **UnknownHostException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **UnknownHostException** * **clone** () const
Clones this exception.
- virtual ~**UnknownHostException** () throw ()

6.711.1 Constructor & Destructor Documentation

6.711.1.1 decaf::net::UnknownHostException::UnknownHostException () throw () [inline]

Default Constructor.

6.711.1.2 `decaf::net::UnknownHostException::UnknownHostException (const Exception & ex) throw () [inline]`

Conversion Constructor from some other Exception.

Parameters

ex An exception that should become this type of Exception

References `decaf::lang::Exception::Exception()`.

6.711.1.3 `decaf::net::UnknownHostException::UnknownHostException (const UnknownHostException & ex) throw () [inline]`

Copy Constructor.

Parameters

ex An exception that should become this type of Exception

References `decaf::lang::Exception::Exception()`.

6.711.1.4 `decaf::net::UnknownHostException::UnknownHostException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.711.1.5 `decaf::net::UnknownHostException::UnknownHostException (const std::exception * cause) throw () [inline]`

Constructor.

Parameters

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.711.1.6 `decaf::net::UnknownHostException::UnknownHostException (const char * file, const int lineNumber, const char * msg, ...) throw ()` [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.711.1.7 `virtual decaf::net::UnknownHostException::~~UnknownHostException () throw ()` [inline, virtual]

6.711.2 Member Function Documentation

6.711.2.1 `virtual UnknownHostException* decaf::net::UnknownHostException::clone () const` [inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

a new Exception instance that is a copy of this Exception object.

Reimplemented from `decaf::io::IOException` (p. 1709).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/UnknownHostException.h`

6.712 decaf::net::UnknownServiceException Class Reference

```
#include <src/main/decaf/net/UnknownServiceException.h>
```

Inheritance diagram for `decaf::net::UnknownServiceException`:

Public Member Functions

- `UnknownServiceException () throw ()`
Default Constructor.

- **UnknownServiceException** (const Exception &ex) throw ()
Conversion Constructor from some other Exception.
- **UnknownServiceException** (const **UnknownServiceException** &ex) throw ()
Copy Constructor.
- **UnknownServiceException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **UnknownServiceException** (const std::exception *cause) throw ()
Constructor.
- **UnknownServiceException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **UnknownServiceException** * **clone** () const
Clones this exception.
- virtual ~**UnknownServiceException** () throw ()

6.712.1 Constructor & Destructor Documentation

6.712.1.1 decaf::net::UnknownServiceException::UnknownServiceException () throw () [inline]

Default Constructor.

6.712.1.2 decaf::net::UnknownServiceException::UnknownServiceException (const Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

Parameters

ex An exception that should become this type of Exception

References decaf::lang::Exception::Exception().

6.712.1.3 decaf::net::UnknownServiceException::UnknownServiceException (const UnknownServiceException & ex) throw () [inline]

Copy Constructor.

Parameters

ex An exception that should become this type of Exception

References decaf::lang::Exception::Exception().

6.712.1.4 `decaf::net::UnknownServiceException::UnknownServiceException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.712.1.5 `decaf::net::UnknownServiceException::UnknownServiceException (const std::exception * cause) throw () [inline]`

Constructor.

Parameters

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.712.1.6 `decaf::net::UnknownServiceException::UnknownServiceException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.712.1.7 `virtual decaf::net::UnknownServiceException::~UnknownServiceException () throw () [inline, virtual]`

6.712.2 Member Function Documentation

6.712.2.1 `virtual UnknownServiceException* decaf::net::UnknownServiceException::clone () const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::io::IOException** (p. 1709).

The documentation for this class was generated from the following file:

- `src/main/decaf/net/UnknownServiceException.h`

6.713 decaf::io::UnsupportedEncodingException Class Reference

Thrown when the the Character Encoding is not supported.

```
#include <src/main/decaf/io/UnsupportedEncodingException.h>
```

Inheritance diagram for decaf::io::UnsupportedEncodingException:

Public Member Functions

- **UnsupportedEncodingException** () throw ()
Default Constructor.
- **UnsupportedEncodingException** (const lang::Exception &ex) throw ()
Copy Constructor.
- **UnsupportedEncodingException** (const **UnsupportedEncodingException** &ex) throw ()
Copy Constructor.
- **UnsupportedEncodingException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **UnsupportedEncodingException** (const std::exception *cause) throw ()
Constructor.
- **UnsupportedEncodingException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor.
- virtual **UnsupportedEncodingException** * clone () const
Clones this exception.
- virtual ~**UnsupportedEncodingException** () throw ()

6.713.1 Detailed Description

Thrown when the the Character Encoding is not supported.

Since

1.0

6.713.2 Constructor & Destructor Documentation

6.713.2.1 `decaf::io::UnsupportedEncodingException::UnsupportedEncodingException () throw () [inline]`

Default Constructor.

6.713.2.2 `decaf::io::UnsupportedEncodingException::UnsupportedEncodingException (const lang::Exception & ex) throw () [inline]`

Copy Constructor.

Parameters

ex the exception to copy

6.713.2.3 `decaf::io::UnsupportedEncodingException::UnsupportedEncodingException (const UnsupportedEncodingException & ex) throw () [inline]`

Copy Constructor.

Parameters

ex the exception to copy, which is an instance of this type

6.713.2.4 `decaf::io::UnsupportedEncodingException::UnsupportedEncodingException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.713.2.5 `decaf::io::UnsupportedEncodingException::UnsupportedEncodingException`
`(const std::exception * cause) throw () [inline]`

Constructor.

Parameters

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.713.2.6 `decaf::io::UnsupportedEncodingException::UnsupportedEncodingException`
`(const char * file, const int lineNumber, const char * msg, ...)`
`throw () [inline]`

Constructor.

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.713.2.7 `virtual`
`decaf::io::UnsupportedEncodingException::~~UnsupportedEncodingException`
`() throw () [inline, virtual]`

6.713.3 Member Function Documentation

6.713.3.1 `virtual UnsupportedEncodingException* de-`
`caf::io::UnsupportedEncodingException::clone () const [inline,`
`virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

A new instance of an Exception object that is a copy of this instance.

Reimplemented from `decaf::io::IOException` (p. 1709).

The documentation for this class was generated from the following file:

- `src/main/decaf/io/UnsupportedEncodingException.h`

6.714 cms::UnsupportedOperationException Class Reference

This exception must be thrown when a CMS client attempts use a CMS method that is not implemented or not supported by the CMS Provider in use.

```
#include <src/main/cms/UnsupportedOperationException.h>
```

Inheritance diagram for cms::UnsupportedOperationException:

Public Member Functions

- **UnsupportedOperationException** () throw ()
- **UnsupportedOperationException** (const **UnsupportedOperationException** &ex) throw ()
- **UnsupportedOperationException** (const std::string &message, const std::exception *cause) throw ()
- **UnsupportedOperationException** (const std::string &message, const std::exception *cause, const std::vector< std::pair< std::string, int > > &stackTrace) throw ()
- virtual ~**UnsupportedOperationException** () throw ()

6.714.1 Detailed Description

This exception must be thrown when a CMS client attempts use a CMS method that is not implemented or not supported by the CMS Provider in use.

Since

2.0

6.714.2 Constructor & Destructor Documentation

- 6.714.2.1** cms::UnsupportedOperationException::UnsupportedOperationException () throw ()
- 6.714.2.2** cms::UnsupportedOperationException::UnsupportedOperationException (const UnsupportedOperationException & *ex*) throw ()
- 6.714.2.3** cms::UnsupportedOperationException::UnsupportedOperationException (const std::string & *message*, const std::exception * *cause*) throw ()
- 6.714.2.4** cms::UnsupportedOperationException::UnsupportedOperationException (const std::string & *message*, const std::exception * *cause*, const std::vector< std::pair< std::string, int > > & *stackTrace*) throw ()
- 6.714.2.5** virtual cms::UnsupportedOperationException::~~UnsupportedOperationException () throw () [virtual]

The documentation for this class was generated from the following file:

- `src/main/cms/UnsupportedOperationException.h`

6.715 decaf::lang::exceptions::UnsupportedOperationException Class Reference

```
#include <src/main/decaf/lang/exceptions/UnsupportedOperationException.h>
```

Inheritance diagram for `decaf::lang::exceptions::UnsupportedOperationException`:

Public Member Functions

- **UnsupportedOperationException** () throw ()
Default Constructor.
- **UnsupportedOperationException** (const **Exception** &ex) throw ()
*Conversion Constructor from some other **Exception** (p. 1476).*
- **UnsupportedOperationException** (const **UnsupportedOperationException** &ex) throw ()
Copy Constructor.
- **UnsupportedOperationException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **UnsupportedOperationException** (const std::exception *cause) throw ()
Constructor.
- **UnsupportedOperationException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **UnsupportedOperationException** * clone () const
Clones this exception.
- virtual ~**UnsupportedOperationException** () throw ()

6.715.1 Constructor & Destructor Documentation

6.715.1.1 decaf::lang::exceptions::UnsupportedOperationException::UnsupportedOperationException () throw () [inline]

Default Constructor.

6.715.1.2 decaf::lang::exceptions::UnsupportedOperationException::UnsupportedOperationException
(const Exception & *ex*) throw () [inline]

Conversion Constructor from some other **Exception** (p. 1476).

Parameters

ex An exception that should become this type of **Exception** (p. 1476)

6.715.1.3 decaf::lang::exceptions::UnsupportedOperationException::UnsupportedOperationException
(const UnsupportedOperationException & *ex*) throw () [inline]

Copy Constructor.

Parameters

ex An exception that should become this type of **Exception** (p. 1476)

6.715.1.4 decaf::lang::exceptions::UnsupportedOperationException::UnsupportedOperationException
(const char * *file*, const int *lineNumber*, const std::exception *
cause, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.715.1.5 decaf::lang::exceptions::UnsupportedOperationException::UnsupportedOperationException
(const std::exception * *cause*) throw () [inline]

Constructor.

Parameters

cause **Pointer** (p. 2343) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.715.1.6 decaf::lang::exceptions::UnsupportedOperationException::UnsupportedOperationException
(const char * *file*, const int *lineNumber*, const char * *msg*, ...)
throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

- file* The file name where exception occurs
- lineNumber* The line number where the exception occurred.
- msg* The message to report
- ... list of primitives that are formatted into the message

6.715.1.7 **virtual**
 `decaf::lang::exceptions::UnsupportedOperationException::~~UnsupportedOperationException`
 `() throw () [inline, virtual]`

6.715.2 Member Function Documentation

6.715.2.1 **virtual UnsupportedOperationException* decaf::lang::exceptions::UnsupportedOperationException::clone () const**
 `[inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

an new **Exception** (p. 1476) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1479).

Reimplemented in **decaf::nio::ReadOnlyBufferException** (p. 2522).

The documentation for this class was generated from the following file:

- `src/main/decaf/lang/exceptions/UnsupportedOperationException.h`

6.716 decaf::net::URI Class Reference

This class represents an instance of a **URI** (p. 3096) as defined by RFC 2396.

```
#include <src/main/decaf/net/URI.h>
```

Inheritance diagram for `decaf::net::URI`:

Public Member Functions

- **URI ()**
 Default Constructor, same as calling a Constructor with all fields empty.
- **URI (const URI &uri) throw (URISyntaxException)**
 *Constructs a **URI** (p. 3096) as a copy of another **URI** (p. 3096).*

- **URI** (const std::string &uri) throw (URISyntaxException)
*Constructs a **URI** (p. 3096) from the given string.*
- **URI** (const std::string &scheme, const std::string &ssp, const std::string &fragment) throw (URISyntaxException)
*Constructs a **URI** (p. 3096) from the given components.*
- **URI** (const std::string &scheme, const std::string &userInfo, const std::string &host, int port, const std::string &path, const std::string &query, const std::string &fragment) throw (URISyntaxException)
*Constructs a **URI** (p. 3096) from the given components.*
- **URI** (const std::string &scheme, const std::string &host, const std::string &path, const std::string &fragment) throw (URISyntaxException)
*Constructs a **URI** (p. 3096) from the given components.*
- **URI** (const std::string &scheme, const std::string &authority, const std::string &path, const std::string &query, const std::string &fragment) throw (URISyntaxException)
*Constructs a **URI** (p. 3096) from the given components.*
- virtual ~**URI** ()
- virtual int **compareTo** (const **URI** &value) const
Compares this object with the specified object for order.
- virtual bool **equals** (const **URI** &value) const
- virtual bool **operator==** (const **URI** &value) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const **URI** &value) const
Compares this object to another and returns true if this object is considered to be less than the one passed.
- std::string **getAuthority** () const
- std::string **getFragment** () const
- std::string **getHost** () const
- std::string **getPath** () const
- int **getPort** () const
- std::string **getQuery** () const
- std::string **getScheme** () const
- std::string **getUserInfo** () const
- std::string **getRawAuthority** () const
*Returns the raw authority component of this **URI** (p. 3096).*
- std::string **getRawFragment** () const
*Returns the raw fragment component of this **URI** (p. 3096).*
- std::string **getRawPath** () const
*Returns the raw path component of this **URI** (p. 3096).*
- std::string **getRawQuery** () const

*Returns the raw query component of this **URI** (p. 3096).*

- `std::string getRawSchemeSpecificPart () const`
*Returns the raw scheme-specific part of this **URI** (p. 3096).*
- `std::string getSchemeSpecificPart () const`
*Returns the decoded scheme-specific part of this **URI** (p. 3096).*
- `std::string getRawUserInfo () const`
*Returns the raw user-information component of this **URI** (p. 3096).*
- `bool isAbsolute () const`
*Tells whether or not this **URI** (p. 3096) is absolute.*
- `bool isOpaque () const`
*Tells whether or not this **URI** (p. 3096) is opaque.*
- `URI normalize () const`
*Normalizes this **URI**'s path.*
- `URI parseServerAuthority () const throw (URISyntaxException)`
*Attempts to parse this **URI**'s authority component, if defined, into user-information, host, and port components.*
- `URI relativize (const URI &uri) const`
*Relativizes the given **URI** (p. 3096) against this **URI** (p. 3096).*
- `URI resolve (const std::string &str) const throw (lang::exceptions::IllegalArgumentException)`
*Constructs a new **URI** (p. 3096) by parsing the given string and then resolving it against this **URI** (p. 3096).*
- `URI resolve (const URI &uri) const`
*Resolves the given **URI** (p. 3096) against this **URI** (p. 3096).*
- `std::string toString () const`
*Returns the content of this **URI** (p. 3096) as a string.*
- `URL toURL () const throw (MalformedURLException, lang::exceptions::IllegalArgumentException)`
*Constructs a **URL** (p. 3133) from this **URI** (p. 3096).*

Static Public Member Functions

- `static URI create (const std::string uri) throw (lang::exceptions::IllegalArgumentException)`
*Creates a **URI** (p. 3096) by parsing the given string.*

6.716.1 Detailed Description

This class represents an instance of a **URI** (p. 3096) as defined by RFC 2396.

6.716.2 Constructor & Destructor Documentation

6.716.2.1 decaf::net::URI::URI ()

Default Constructor, same as calling a Constructor with all fields empty.

6.716.2.2 decaf::net::URI::URI (const URI & *uri*) throw (URISyntaxException)

Constructs a **URI** (p. 3096) as a copy of another **URI** (p. 3096).

Parameters

uri - uri to copy

6.716.2.3 decaf::net::URI::URI (const std::string & *uri*) throw (URISyntaxException)

Constructs a **URI** (p. 3096) from the given string.

Parameters

uri - string uri to parse.

6.716.2.4 decaf::net::URI::URI (const std::string & *scheme*, const std::string & *ssp*, const std::string & *fragment*) throw (URISyntaxException)

Constructs a **URI** (p. 3096) from the given components.

Parameters

scheme - the uri scheme

ssp - Scheme specific part

fragment - Fragment

6.716.2.5 decaf::net::URI::URI (const std::string & *scheme*, const std::string & *userInfo*, const std::string & *host*, int *port*, const std::string & *path*, const std::string & *query*, const std::string & *fragment*) throw (URISyntaxException)

Constructs a **URI** (p. 3096) from the given components.

Parameters

scheme - Scheme name

userInfo - User name and authorization information

host - Host name

port - Port number

path - Path

query - Query

fragment - Fragment

6.716.2.6 `decaf::net::URI::URI (const std::string & scheme, const std::string & host, const std::string & path, const std::string & fragment) throw (URISyntaxException)`

Constructs a **URI** (p. 3096) from the given components.

Parameters

scheme - Scheme name

host - Host name

path - Path

fragment - Fragment

6.716.2.7 `decaf::net::URI::URI (const std::string & scheme, const std::string & authority, const std::string & path, const std::string & query, const std::string & fragment) throw (URISyntaxException)`

Constructs a **URI** (p. 3096) from the given components.

Parameters

scheme - Scheme name

authority - Authority

path - Path

query - Query

fragment - Fragment

6.716.2.8 `virtual decaf::net::URI::~~URI () [inline, virtual]`

6.716.3 Member Function Documentation

6.716.3.1 `virtual int decaf::net::URI::compareTo (const URI & value) const [virtual]`

Compares this object with the specified object for order.

Returns a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

Parameters

value - the value to compare to this one.

Returns

zero if equal minus one if less than and one if greater than.

6.716.3.2 static URI decaf::net::URI::create (const std::string uri) throw (lang::exceptions::IllegalArgumentException) [static]

Creates a **URI** (p. 3096) by parsing the given string.

This convenience factory method works as if by invoking the `URI(string)` constructor; any **URISyntaxException** (p. 3122) thrown by the constructor is caught and wrapped in a new `IllegalArgumentException` object, which is then thrown.

Parameters

uri - **URI** (p. 3096) string to parse

Exceptions

IllegalArgumentException

6.716.3.3 virtual bool decaf::net::URI::equals (const URI & value) const [virtual]**Returns**

true if this value is considered equal to the passed value.

6.716.3.4 std::string decaf::net::URI::getAuthority () const**Returns**

the decoded authority component of this **URI** (p. 3096).

6.716.3.5 std::string decaf::net::URI::getFragment () const**Returns**

the decoded fragment component of this **URI** (p. 3096).

6.716.3.6 std::string decaf::net::URI::getHost () const**Returns**

the host component of this **URI** (p. 3096).

6.716.3.7 `std::string decaf::net::URI::getPath () const`**Returns**

the path component of this **URI** (p. 3096).

6.716.3.8 `int decaf::net::URI::getPort () const`**Returns**

the port component of this **URI** (p. 3096).

6.716.3.9 `std::string decaf::net::URI::getQuery () const`**Returns**

the query component of this **URI** (p. 3096).

6.716.3.10 `std::string decaf::net::URI::getRawAuthority () const`

Returns the raw authority component of this **URI** (p. 3096).

The authority component of a **URI** (p. 3096), if defined, only contains the commercial-at character ('@') and characters in the unreserved, punct, escaped, and other categories. If the authority is server-based then it is further constrained to have valid user-information, host, and port components.

Returns

the raw authority component of the **URI** (p. 3096)

6.716.3.11 `std::string decaf::net::URI::getRawFragment () const`

Returns the raw fragment component of this **URI** (p. 3096).

The fragment component of a **URI** (p. 3096), if defined, only contains legal **URI** (p. 3096) characters.

Returns

the raw fragment component of this **URI** (p. 3096)

6.716.3.12 `std::string decaf::net::URI::getRawPath () const`

Returns the raw path component of this **URI** (p. 3096).

The path component of a **URI** (p. 3096), if defined, only contains the slash character ('/'), the commercial-at character ('@'), and characters in the unreserved, punct, escaped, and other categories.

Returns

the raw path component of this **URI** (p. 3096)

6.716.3.13 `std::string decaf::net::URI::getRawQuery () const`

Returns the raw query component of this **URI** (p. 3096).

The query component of a **URI** (p. 3096), if defined, only contains legal **URI** (p. 3096) characters.

Returns

the raw query component of the **URI** (p. 3096).

6.716.3.14 `std::string decaf::net::URI::getRawSchemeSpecificPart () const`

Returns the raw scheme-specific part of this **URI** (p. 3096).

The scheme-specific part is never undefined, though it may be empty. The scheme-specific part of a **URI** (p. 3096) only contains legal **URI** (p. 3096) characters.

Returns

the raw scheme special part of the uri

6.716.3.15 `std::string decaf::net::URI::getRawUserInfo () const`

Returns the raw user-information component of this **URI** (p. 3096).

The user-information component of a **URI** (p. 3096), if defined, only contains characters in the unreserved, punct, escaped, and other categories.

Returns

the raw user-information component of the **URI** (p. 3096)

6.716.3.16 `std::string decaf::net::URI::getScheme () const`**Returns**

the scheme component of this **URI** (p. 3096)

6.716.3.17 `std::string decaf::net::URI::getSchemeSpecificPart () const`

Returns the decoded scheme-specific part of this **URI** (p. 3096).

The string returned by this method is equal to that returned by the `getRawSchemeSpecificPart` method except that all sequences of escaped octets are decoded.

Returns

the raw scheme specific part of the uri.

6.716.3.18 `std::string decaf::net::URI::getUserInfo () const`**Returns**

the user info component of this **URI** (p. 3096)

6.716.3.19 `bool decaf::net::URI::isAbsolute () const`

Tells whether or not this **URI** (p. 3096) is absolute.

A **URI** (p. 3096) is absolute if, and only if, it has a scheme component.

Returns

true if, and only if, this **URI** (p. 3096) is absolute

6.716.3.20 `bool decaf::net::URI::isOpaque () const`

Tells whether or not this **URI** (p. 3096) is opaque.

A **URI** (p. 3096) is opaque if, and only if, it is absolute and its scheme-specific part does not begin with a slash character ('/'). An opaque **URI** (p. 3096) has a scheme, a scheme-specific part, and possibly a fragment; all other components are undefined.

Returns

true if, and only if, this **URI** (p. 3096) is opaque

6.716.3.21 `URI decaf::net::URI::normalize () const`

Normalizes this **URI**'s path.

If this **URI** (p. 3096) is opaque, or if its path is already in normal form, then this **URI** (p. 3096) is returned. Otherwise a new **URI** (p. 3096) is constructed that is identical to this **URI** (p. 3096) except that its path is computed by normalizing this **URI**'s path in a manner consistent with RFC 2396, section 5.2, step 6, sub-steps c through f; that is:

1. All "." segments are removed. 2. If a ".." segment is preceded by a non-"" segment then both of these segments are removed. This step is repeated until it is no longer applicable. 3. If the path is relative, and if its first segment contains a colon character (':'), then a "." segment is prepended. This prevents a relative **URI** (p. 3096) with a path such as "a:b/c/d" from later being re-parsed as an opaque **URI** (p. 3096) with a scheme of "a" and a scheme-specific part of "b/c/d". (Deviation from RFC 2396)

A normalized path will begin with one or more "." segments if there were insufficient non-"" segments preceding them to allow their removal. A normalized path will begin with a "." segment if one was inserted by step 3 above. Otherwise, a normalized path will not contain any "." or ".." segments.

Returns

A **URI** (p. 3096) equivalent to this **URI** (p. 3096), but whose path is in normal form

**6.716.3.22 `virtual bool decaf::net::URI::operator< (const URI & value) const`
[virtual]**

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

Parameters

value - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

6.716.3.23 `virtual bool decaf::net::URI::operator==(const URI & value) const`
[virtual]

Compares equality between this object and the one passed.

Parameters

value - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

6.716.3.24 `URI decaf::net::URI::parseServerAuthority () const throw (`
`URISyntaxException)`

Attempts to parse this URI's authority component, if defined, into user-information, host, and port components.

If this URI's authority component has already been recognized as being server-based then it will already have been parsed into user-information, host, and port components. In this case, or if this **URI** (p. 3096) has no authority component, this method simply returns this **URI** (p. 3096).

Otherwise this method attempts once more to parse the authority component into user-information, host, and port components, and throws an exception describing why the authority component could not be parsed in that way.

Returns

A **URI** (p. 3096) whose authority field has been parsed as a server-based authority

Exceptions

URISyntaxException (p. 3122) - If the authority component of this **URI** (p. 3096) is defined but cannot be parsed as a server-based authority.

6.716.3.25 `URI decaf::net::URI::relativize (const URI & uri) const`

Relativizes the given **URI** (p. 3096) against this **URI** (p. 3096).

The relativization of the given **URI** (p. 3096) against this **URI** (p. 3096) is computed as follows:

1. If either this **URI** (p. 3096) or the given **URI** (p. 3096) are opaque, or if the scheme and authority components of the two URIs are not identical, or if the path of this **URI** (p. 3096) is not a prefix of the path of the given **URI** (p. 3096), then the given **URI** (p. 3096) is returned.
2. Otherwise a new relative hierarchical **URI** (p. 3096) is constructed with query and fragment components taken from the given **URI** (p. 3096) and with a path component computed by removing this URI's path from the beginning of the given URI's path.

Parameters

uri - The **URI** (p. 3096) to be relativized against this **URI** (p. 3096)

Returns

The resulting **URI** (p. 3096)

6.716.3.26 URI decaf::net::URI::resolve (const std::string & *str*) const throw (lang::exceptions::IllegalArgumentException)

Constructs a new **URI** (p. 3096) by parsing the given string and then resolving it against this **URI** (p. 3096).

This convenience method works as if invoking it were equivalent to evaluating the expression `resolve(URI::create(str))`.

Parameters

str - The string to be parsed into a **URI** (p. 3096)

Returns

The resulting **URI** (p. 3096)

Exceptions

IllegalArgumentException - If the given string violates RFC 2396

6.716.3.27 URI decaf::net::URI::resolve (const URI & *uri*) const

Resolves the given **URI** (p. 3096) against this **URI** (p. 3096).

If the given **URI** (p. 3096) is already absolute, or if this **URI** (p. 3096) is opaque, then a copy of the given **URI** (p. 3096) is returned.

If the given URI's fragment component is defined, its path component is empty, and its scheme, authority, and query components are undefined, then a **URI** (p. 3096) with the given fragment but with all other components equal to those of this **URI** (p. 3096) is returned. This allows a **URI** (p. 3096) representing a standalone fragment reference, such as "#foo", to be usefully resolved against a base **URI** (p. 3096).

Otherwise this method constructs a new hierarchical **URI** (p. 3096) in a manner consistent with RFC 2396, section 5.2; that is:

1. A new **URI** (p. 3096) is constructed with this URI's scheme and the given URI's query and fragment components.
2. If the given **URI** (p. 3096) has an authority component then the new URI's authority and path are taken from the given **URI** (p. 3096).
3. Otherwise the new URI's authority component is copied from this **URI** (p. 3096), and its path is computed as follows:

1. If the given URI's path is absolute then the new URI's path is taken from the given **URI** (p. 3096).
2. Otherwise the given URI's path is relative, and so the new URI's path is computed by resolving the path of the given **URI** (p. 3096) against the path of this **URI** (p. 3096). This is done by concatenating all but the last segment of this URI's path, if any, with the given URI's path and then normalizing the result as if by invoking the `normalize` method.

The result of this method is absolute if, and only if, either this **URI** (p. 3096) is absolute or the given **URI** (p. 3096) is absolute.

Parameters

uri - The **URI** (p. 3096) to be resolved against this **URI** (p. 3096)

Returns

The resulting **URI** (p. 3096)

6.716.3.28 std::string decaf::net::URI::toString () const

Returns the content of this **URI** (p. 3096) as a string.

If this **URI** (p. 3096) was created by invoking one of the constructors in this class then a string equivalent to the original input string, or to the string computed from the originally-given components, as appropriate, is returned. Otherwise this **URI** (p. 3096) was created by normalization, resolution, or relativization, and so a string is constructed from this **URI**'s components according to the rules specified in RFC 2396, section 5.2, step 7.

Returns

the string form of this **URI** (p. 3096)

6.716.3.29 URL decaf::net::URI::toURL () const throw (MalformedURLException, lang::exceptions::IllegalArgumentException)

Constructs a **URL** (p. 3133) from this **URI** (p. 3096).

This convenience method works as if invoking it were equivalent to evaluating the expression `new URL (p. 3133)(this.toString())` after first checking that this **URI** (p. 3096) is absolute.

Returns

A **URL** (p. 3133) constructed from this **URI** (p. 3096)

Exceptions

IllegalArgumentException - If this **URL** (p. 3133) is not absolute

MalformedURLException (p. 1968) - If a protocol handler for the **URL** (p. 3133) could not be found, or if some other error occurred while constructing the **URL** (p. 3133)

The documentation for this class was generated from the following file:

- `src/main/decaf/net/URI.h`

6.717 decaf::internal::net::URIEncoderDecoder Class Reference

```
#include <src/main/decaf/internal/net/URIEncoderDecoder.h>
```

Public Member Functions

- **URIEncoderDecoder** ()
- virtual **~URIEncoderDecoder** ()

Static Public Member Functions

- static void **validate** (const std::string &s, const std::string &legal) throw (decaf::net::URISyntaxException)

Validate a string by checking if it contains any characters other than:

- static void **validateSimple** (const std::string &s, const std::string &legal) throw (decaf::net::URISyntaxException)

Validate a string by checking if it contains any characters other than:

- static std::string **quoteIllegal** (const std::string &s, const std::string &legal)

All characters except letters ('a'..'z', 'A'..'Z') and numbers ('0'..'9') and legal characters are converted into their hexadecimal value prepended by ".

- static std::string **encodeOthers** (const std::string &s)

Other characters, which are chars that are not US-ASCII, and are not ISO Control or are not ISO Space chars are not preserved.

- static std::string **decode** (const std::string &s)

Decodes the string argument which is assumed to be encoded in the x-www-form-urlencoded MIME content type using the UTF-8 encoding scheme.

6.717.1 Constructor & Destructor Documentation

6.717.1.1 decaf::internal::net::URIEncoderDecoder::URIEncoderDecoder ()

6.717.1.2 virtual decaf::internal::net::URIEncoderDecoder::~~URIEncoderDecoder () [inline, virtual]

6.717.2 Member Function Documentation

6.717.2.1 static std::string decaf::internal::net::URIEncoderDecoder::decode (const std::string & s) [static]

Decodes the string argument which is assumed to be encoded in the x-www-form-urlencoded MIME content type using the UTF-8 encoding scheme.

" and two following hex digit characters are converted to the equivalent byte value. All other characters are passed through unmodified.

e.g. "A%20B%20C %24%25" -> "A B C \$%"

Parameters

s - The encoded string.

Returns

The decoded version.

6.717.2.2 static std::string decaf::internal::net::URIEncoderDecoder::encodeOthers (const std::string & *s*) [static]

Other characters, which are chars that are not US-ASCII, and are not ISO Control or are not ISO Space chars are not preserved.

They are converted into their hexadecimal value prepended by ”.

For example: Euro currency symbol -> "%E2%82%AC".

Parameters

s - the string to be converted

Returns

the converted string

6.717.2.3 static std::string decaf::internal::net::URIEncoderDecoder::quoteIllegal (const std::string & *s*, const std::string & *legal*) [static]

All characters except letters ('a'..'z', 'A'..'Z') and numbers ('0'..'9') and legal characters are converted into their hexadecimal value prepended by ”.

For example: '#' -> 23

Other characters, which are chars that are not US-ASCII, and are not ISO Control or are not ISO Space chars, are preserved.

Parameters

s - the string to be converted

legal - the characters allowed to be preserved in the string *s*

Returns

converted string

6.717.2.4 static void decaf::internal::net::URIEncoderDecoder::validate (const std::string & *s*, const std::string & *legal*) throw (decaf::net::URISyntaxException) [static]

Validate a string by checking if it contains any characters other than:

1. letters ('a'..'z', 'A'..'Z') 2. numbers ('0'..'9') 3. characters in the legalset parameter 4. characters that are not ISO Control or are not ISO Space characters)

Parameters

s - the string to be validated

legal - the characters allowed in the string *s*

6.717.2.5 `static void decaf::internal::net::URIEncoderDecoder::validateSimple (const std::string & s, const std::string & legal) throw (decaf::net::URISyntaxException) [static]`

Validate a string by checking if it contains any characters other than:

1. letters ('a'..'z', 'A'..'Z')
2. numbers ('0'..'9')
3. characters in the legalset parameter

Parameters

- s* - the string to be validated
- legal* - the characters allowed in the string *s*

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/net/URIEncoderDecoder.h`

6.718 decaf::internal::net::URIHelper Class Reference

Helper class used by the URI classes in encoding and decoding of URI's.

```
#include <src/main/decaf/internal/net/URIHelper.h>
```

Public Member Functions

- **URIHelper** (const std::string &unreserved, const std::string &punct, const std::string &reserved, const std::string &someLegal, const std::string &allLegal)
*Setup the **URIHelper** (p. 3110) with values assigned to the various fields that are used in the validation process.*
- **URIHelper** ()
Sets up the filter strings with sane defaults.
- virtual **~URIHelper** ()
- **URIType** **parseURI** (const std::string &uri, bool forceServer) throw (decaf::net::URISyntaxException)
Parse the passed in URI.
- void **validateScheme** (const std::string &uri, const std::string &scheme, int index) throw (decaf::net::URISyntaxException)
Validate the schema portin of the URI.
- void **validateSsp** (const std::string &uri, const std::string &ssp, std::size_t index) throw (decaf::net::URISyntaxException)
Validate that the URI Ssp Segment contains no invalid encodings.
- void **validateAuthority** (const std::string &uri, const std::string &authority, std::size_t index) throw (decaf::net::URISyntaxException)
Validate that the URI Authority Segment contains no invalid encodings.

- void **validatePath** (const std::string &uri, const std::string &path, std::size_t index) throw (decaf::net::URISyntaxException)

Validate that the URI Path Segment contains no invalid encodings.

- void **validateQuery** (const std::string &uri, const std::string &query, std::size_t index) throw (decaf::net::URISyntaxException)

Validate that the URI Query Segment contains no invalid encodings.

- void **validateFragment** (const std::string &uri, const std::string &fragment, std::size_t index) throw (decaf::net::URISyntaxException)

Validate that the URI fragment contains no invalid encodings.

- **URIType parseAuthority** (bool forceServer, const std::string &authority) throw (decaf::net::URISyntaxException)

determine the host, port and user-info if the authority parses successfully to a server based authority

- void **validateUserinfo** (const std::string &uri, const std::string &userinfo, std::size_t index) throw (decaf::net::URISyntaxException)

Check the supplied user info for validity.

- bool **isValidHost** (bool forceServer, const std::string &host) throw (decaf::net::URISyntaxException)

distinguish between IPv4, IPv6, domain name and validate it based on its type

- bool **isValidDomainName** (const std::string &host)

Validates the string past to determine if it is a well formed domain name.

- bool **isValidIPv4Address** (const std::string &host)

Validate if the host value is a well formed IPv4 address, this is the form XXX.XXX.XXX.XXX were X is any number 0-9.

- bool **isValidIPv6Address** (const std::string &ipAddress)

Determines if the given address is valid according to the IPv6 spec.

- bool **isValidIPv4Word** (const std::string &word)

Check is the string passed contains a Valid IPv4 word, which is an integer in the range of 0 to 255.

- bool **isValidHexChar** (char c)

Determines if the given char is a valid Hex char.

6.718.1 Detailed Description

Helper class used by the URI classes in encoding and decoding of URI's.

6.718.2 Constructor & Destructor Documentation

6.718.2.1 `decaf::internal::net::URIHelper::URIHelper (const std::string & unreserved, const std::string & punct, const std::string & reserved, const std::string & someLegal, const std::string & allLegal)`

Setup the **URIHelper** (p.3110) with values assigned to the various fields that are used in the validation process.

The defaults are overridden by these values.

Parameters

unreserved - characters not reserved for use.

punct - allowable punctuation symbols.

reserved - characters not allowed for general use in the URI.

someLegal - characters that are legal in certain cases.

allLegal - characters that are always legal.

6.718.2.2 `decaf::internal::net::URIHelper::URIHelper ()`

Sets up the filter strings with sane defaults.

6.718.2.3 `virtual decaf::internal::net::URIHelper::~~URIHelper () [inline, virtual]`

6.718.3 Member Function Documentation

6.718.3.1 `bool decaf::internal::net::URIHelper::isValidDomainName (const std::string & host)`

Validates the string past to determine if it is a well formed domain name.

Parameters

host - domain name to validate.

Returns

true if host is well formed.

6.718.3.2 `bool decaf::internal::net::URIHelper::isValidHexChar (char c)`

Determines if the given char is a valid Hex char.

Valid chars are A-F (upper or lower case) and 0-9.

Parameters

c - char to inspect

Returns

true if c is a valid hex char.

6.718.3.3 bool decaf::internal::net::URIHelper::isValidHost (bool *forceServer*, const std::string & *host*) throw (decaf::net::URISyntaxException)

distinguish between IPv4, IPv6, domain name and validate it based on its type

Parameters

forceServer - true if the forceServer mode should be active.

host - Host string to validate.

Returns

true if the host value is a valid domain name.

Exceptions

URISyntaxException if the host is invalid and forceServer is true.

6.718.3.4 bool decaf::internal::net::URIHelper::isValidIP4Word (const std::string & *word*)

Check if the string passed contains a Valid IPv4 word, which is an integer in the range of 0 to 255.

Parameters

word - string value to check.

Returns

true if the word is a valid IPv4 word.

6.718.3.5 bool decaf::internal::net::URIHelper::isValidIP6Address (const std::string & *ipAddress*)

Determines if the given address is valid according to the IPv6 spec.

Parameters

ipAddress - string ip address value to validate.

Returns

true if the address string is valid.

6.718.3.6 bool decaf::internal::net::URIHelper::isValidIPv4Address (const std::string & *host*)

Validate if the host value is a well formed IPv4 address, this is the form XXX.XXX.XXX.XXX where X is any number 0-9.

and XXX is not greater than 255.

Parameters

host - IPv4 address string to parse.

Returns

true if host is a well formed IPv4 address.

6.718.3.7 `URIType decaf::internal::net::URIHelper::parseAuthority (bool forceServer, const std::string & authority) throw (decaf::net::URISyntaxException)`

determine the host, port and user-info if the authority parses successfully to a server based authority

behavior in error cases: if *forceServer* is true, throw `URISyntaxException` with the proper diagnostic messages. if *forceServer* is false assume this is a registry based uri, and just return leaving the host, port and user-info fields undefined.

and there are some error cases where `URISyntaxException` is thrown regardless of the *forceServer* parameter e.g. mal-formed ipv6 address

Parameters

forceServer

authority

Returns

a `URIType` (p. 3126) instance containing the parsed data.

Exceptions

URISyntaxException

6.718.3.8 `URIType decaf::internal::net::URIHelper::parseURI (const std::string & uri, bool forceServer) throw (decaf::net::URISyntaxException)`

Parse the passed in URI.

Parameters

uri - the URI to Parse

forceServer - if true invalid URI data throws an Exception

Returns

a `URIType` (p. 3126) instance containing the parsed data.

Exceptions

URISyntaxException if *forceServer* is true and the URI is invalid.

6.718.3.9 `void decaf::internal::net::URIHelper::validateAuthority (const
std::string & uri, const std::string & authority, std::size_t index)
throw (decaf::net::URISyntaxException)`

Validate that the URI Authority Segment contains no invalid encodings.

Parameters

uri - the full uri.

authority - the Authority to check.

index - position in the uri where Authority starts.

Exceptions

URISyntaxException if the fragment has errors.

6.718.3.10 `void decaf::internal::net::URIHelper::validateFragment (const
std::string & uri, const std::string & fragment, std::size_t index)
throw (decaf::net::URISyntaxException)`

Validate that the URI fragment contains no invalid encodings.

Parameters

uri - the full uri.

fragment - the fragment to check.

index - position in the uri where fragment starts.

Exceptions

URISyntaxException if the fragment has errors.

6.718.3.11 `void decaf::internal::net::URIHelper::validatePath (const std::string
& uri, const std::string & path, std::size_t index) throw (decaf::net::URISyntaxException)`

Validate that the URI Path Segment contains no invalid encodings.

Parameters

uri - the full uri.

path - the path to check.

index - position in the uri where path starts.

Exceptions

URISyntaxException if the fragment has errors.

6.718.3.12 `void decaf::internal::net::URIHelper::validateQuery (const std::string & uri, const std::string & query, std::size_t index) throw (decaf::net::URISyntaxException)`

Validate that the URI Query Segment contains no invalid encodings.

Parameters

uri - the full uri.

query - the query to check.

index - position in the uri where fragment starts.

Exceptions

URISyntaxException if the fragment has errors.

6.718.3.13 `void decaf::internal::net::URIHelper::validateScheme (const std::string & uri, const std::string & scheme, int index) throw (decaf::net::URISyntaxException)`

Validate the schema portin of the URI.

Parameters

uri - the URI to check.

scheme - the schema section of the URI.

index - index in uri where schema starts.

Exceptions

URISyntaxException if the fragment has errors.

6.718.3.14 `void decaf::internal::net::URIHelper::validateSsp (const std::string & uri, const std::string & ssp, std::size_t index) throw (decaf::net::URISyntaxException)`

Validate that the URI Ssp Segment contains no invalid encodings.

Parameters

uri - the full uri.

ssp - the SSP to check.

index - position in the uri where Ssp starts.

Exceptions

URISyntaxException if the fragment has errors.


```
6.718.3.15 void decaf::internal::net::URIHelper::validateUserinfo ( const
std::string & uri, const std::string & userinfo, std::size_t index )
throw ( decaf::net::URISyntaxException )
```

Check the supplied user info for validity.

Parameters

uri - the uri to parse.

userinfo - supplied user info

index - index into the URI string where the data is located.

Returns

true if valid

Exceptions

URISyntaxException if an error occurs

The documentation for this class was generated from the following file:

- src/main/decaf/internal/net/URIHelper.h

6.719 activemq::transport::failover::URIPool Class Reference

```
#include <src/main/activemq/transport/failover/URIPool.h>
```

Public Member Functions

- **URIPool** ()
Create an Empty URI Pool.
- **URIPool** (const decaf::util::List< URI > &uris)
Creates a new URI Pool using the given list as the initial Free List.
- virtual ~**URIPool** ()
- **URI** **getURI** () throw (decaf::lang::exceptions::NoSuchElementException)
Fetches the next available URI from the pool, if there are no more URIs free when this method is called it throws a NoSuchElementException.
- void **addURI** (const URI &uri)
*Adds a URI to the free list, callers that have previously taken one using the **getURI** method should always return the URI when they close the resource that was connected to that URI.*
- void **addURIs** (const StlList< URI > &uris)
Adds a List of URIs to this Pool, the method checks for duplicates already in the pool and does not add those.

- **void removeURI** (const **URI** &uri)
Remove a given URI from the Free List.
- **bool isRandomize** () const
Is the URI that is given randomly picked from the pool or is each one taken in sequence.
- **void setRandomize** (bool value)
Sets if the URI's that are taken from the pool are chosen Randomly or are taken in the order they are in the list.

6.719.1 Constructor & Destructor Documentation

6.719.1.1 **activemq::transport::failover::URIPool::URIPool** ()

Create an Empty URI Pool.

6.719.1.2 **activemq::transport::failover::URIPool::URIPool** (const **decaf::util::List**< **URI** > & *uris*)

Creates a new URI Pool using the given list as the initial Free List.

Parameters

uris - List of URI to place in the Pool.

6.719.1.3 **virtual activemq::transport::failover::URIPool::~~URIPool** () [virtual]

6.719.2 Member Function Documentation

6.719.2.1 **void activemq::transport::failover::URIPool::addURI** (const **URI** & *uri*)

Adds a URI to the free list, callers that have previously taken one using the **getURI** method should always return the URI when they close the resource that was connected to that URI.

Parameters

uri - a URI previously taken from the pool.

6.719.2.2 **void activemq::transport::failover::URIPool::addURIs** (const **StlList**< **URI** > & *uris*)

Adds a List of URIs to this Pool, the method checks for duplicates already in the pool and does not add those.

Parameters

uris - List of URIs to add into the Pool.

6.719.2.3 URI activemq::transport::failover::URIPool::getURI () throw (decaf::lang::exceptions::NoSuchElementException)

Fetches the next available URI from the pool, if there are no more URIs free when this method is called it throws a `NoSuchElementException`.

Receiving the exception is not an indication that a URI won't be available in the future, the caller should react accordingly.

Returns

the next free URI in the Pool.

Exceptions

NoSuchElementException if there are none free currently.

6.719.2.4 bool activemq::transport::failover::URIPool::isRandomize () const [inline]

Is the URI that is given randomly picked from the pool or is each one taken in sequence.

Returns

true if URI gets are random.

6.719.2.5 void activemq::transport::failover::URIPool::removeURI (const URI & uri)

Remove a given URI from the Free List.

Parameters

uri - the URI to find and remove from the free list

6.719.2.6 void activemq::transport::failover::URIPool::setRandomize (bool value) [inline]

Sets if the URIs that are taken from the pool are chosen Randomly or are taken in the order they are in the list.

Parameters

value - true indicates URI gets are random.

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/failover/URIPool.h`

6.720 activemq::util::URISupport Class Reference

```
#include <src/main/activemq/util/URISupport.h>
```

Static Public Member Functions

- static void **parseURL** (const std::string &URI, **decaf::util::Properties** &properties) throw (decaf::lang::exceptions::IllegalArgumentException)

Parses the properties out of the provided Broker URI and sets them in the passed Properties Object.

- static **CompositeData** **parseComposite** (const **URI** &uri) throw (decaf::net::URISyntaxException)

Parses a Composite URI into a Composite Data instance, the Composite URI takes the for scheme://(uri1,uri2,...uriN)?param1=value1, each of the composite URIs is stored in the CompositeData's internal list.

- static **decaf::util::Properties** **parseQuery** (std::string query) throw (decaf::lang::exceptions::IllegalArgumentException)

Parse the Query portion of a URI String and return a Simple Properties object containing the parameter names as keys, and the parameter values and values of the Properties.

- static void **parseQuery** (std::string query, **decaf::util::Properties** *properties) throw (decaf::lang::exceptions::IllegalArgumentException)

Parse the Query portion of a URI String and return a Simple Properties object containing the parameter names as keys, and the parameter values and values of the Properties.

- static std::string **createQueryString** (const **Properties** &options) throw (decaf::net::URISyntaxException)

Given a properties object create a string that can be appended to a URI as a valid Query string.

6.720.1 Member Function Documentation

- ### 6.720.1.1 static std::string activemq::util::URISupport::createQueryString (const **Properties** & *options*) throw (decaf::net::URISyntaxException) [static]

Given a properties object create a string that can be appended to a URI as a valid Query string.

Parameters

options Properties object containing key / value query values.

Returns

a valid URI query string.

Exceptions

URISyntaxException if the string in the Properties object can't be encoded into a valid URI Query string.

6.720.1.2 static `CompositeData` `activemq::util::URISupport::parseComposite` (`const URI & uri`) throw (`decaf::net::URISyntaxException`) [static]

Parses a Composite URI into a Composite Data instance, the Composite URI takes the for scheme://(uri1,uri2,...uriN)?param1=value1, each of the composite URIs is stored in the CompositeData's internal list.

Parameters

uri - The Composite URI to parse.

Returns

a new `CompositeData` (p. 1013) object with the parsed data

Exceptions

URISyntaxException if the URI is not well formed.

6.720.1.3 static void `activemq::util::URISupport::parseQuery` (`std::string query`, `decaf::util::Properties * properties`) throw (`decaf::lang::exceptions::IllegalArgumentException`) [static]

Parse the Query portion of a URI String and return a Simple Properties object containing the parameter names as keys, and the parameter values and values of the Properties.

Parameters

query - the query string to parse.

properties - object pointer to get the parsed output.

Exceptions

IllegalArgumentException if the Query string is not well formed.

6.720.1.4 static `decaf::util::Properties` `activemq::util::URISupport::parseQuery` (`std::string query`) throw (`decaf::lang::exceptions::IllegalArgumentException`) [static]

Parse the Query portion of a URI String and return a Simple Properties object containing the parameter names as keys, and the parameter values and values of the Properties.

Parameters

query The query string to parse and extract the encoded properties.

Returns

Properties object with the parsed output.

Exceptions

IllegalArgumentException if the Query string is not well formed.

6.720.1.5 `static void activemq::util::URISupport::parseURL (const
std::string & URI, decaf::util::Properties & properties) throw (
decaf::lang::exceptions::IllegalArgumentException) [static]`

Parses the properties out of the provided Broker URI and sets them in the passed Properties Object.

Parameters

URI a Broker URI to parse

properties a Properties object to set the parsed values in

Exceptions

IllegalArgumentException if the passed URI is invalid

The documentation for this class was generated from the following file:

- `src/main/activemq/util/URISupport.h`

6.721 decaf::net::URISyntaxException Class Reference

`#include <src/main/decaf/net/URISyntaxException.h>`

Inheritance diagram for `decaf::net::URISyntaxException`:

Public Member Functions

- **URISyntaxException** () throw ()
Default Constructor.
- **URISyntaxException** (const Exception &ex) throw ()
Conversion Constructor from some other Exception.
- **URISyntaxException** (const **URISyntaxException** &ex) throw ()
Copy Constructor.
- **URISyntaxException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **URISyntaxException** (const std::exception *cause) throw ()
Constructor.
- **URISyntaxException** (const char *file, const int lineNumber, const char *msg DECAF_ -
UNUSED) throw ()
Constructor - Initializes the file name and line number where this message occurred.

- **URISyntaxException** (const char *file, const int lineNumber, const std::string &input, const std::string &reason) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **URISyntaxException** (const char *file, const int lineNumber, const std::string &input, const std::string &reason, std::size_t index) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- virtual **URISyntaxException * clone** () const
Clones this exception.
- virtual **~URISyntaxException** () throw ()
- std::string **getInput** () const
- std::string **getReason** () const
- std::size_t **getIndex** () const

6.721.1 Constructor & Destructor Documentation

6.721.1.1 decaf::net::URISyntaxException::URISyntaxException () throw () [inline]

Default Constructor.

6.721.1.2 decaf::net::URISyntaxException::URISyntaxException (const Exception & ex) throw () [inline]

Conversion Constructor from some other Exception.

Parameters

ex An exception that should become this type of Exception

6.721.1.3 decaf::net::URISyntaxException::URISyntaxException (const URISyntaxException & ex) throw () [inline]

Copy Constructor.

Parameters

ex An exception that should become this type of Exception

6.721.1.4 decaf::net::URISyntaxException::URISyntaxException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs
lineNumber The line number where the exception occurred.
cause The exception that was the cause for this one to be thrown.
msg The message to report
... list of primitives that are formatted into the message

6.721.1.5 `decaf::net::URISyntaxException::URISyntaxException (const std::exception * cause) throw () [inline]`

Constructor.

Parameters

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.721.1.6 `decaf::net::URISyntaxException::URISyntaxException (const char * file, const int lineNumber, const char *msg DECAF_UNUSED) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs
lineNumber The line number where the exception occurred.
msg The message to report
... list of primitives that are formatted into the message

6.721.1.7 `decaf::net::URISyntaxException::URISyntaxException (const char * file, const int lineNumber, const std::string & input, const std::string & reason) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the input string that caused the error and the reason for the error.

Parameters

file The file name where exception occurs.
lineNumber The line number where the exception occurred.
input The **URL** (p.3133) that caused the exception.
reason The reason for the failure.

6.721.1.8 `decaf::net::URISyntaxException::URISyntaxException (const char *
file, const int lineNumber, const std::string & input, const std::string
& reason, std::size_t index) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the input string that caused the error and the reason for the error.

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

input The input **URI** (p. 3096) that caused the exception

reason The reason for the failure.

index The index in the **URI** (p. 3096) string where the error occurred.

6.721.1.9 `virtual decaf::net::URISyntaxException::~~URISyntaxException ()
throw () [inline, virtual]`

6.721.2 Member Function Documentation

6.721.2.1 `virtual URISyntaxException* decaf::net::URISyntaxException::clone (
) const [inline, virtual]`

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::lang::Exception** (p. 1479).

6.721.2.2 `std::size_t decaf::net::URISyntaxException::getIndex () const
[inline]`

Returns

the index in the input string where the error occurred or -1

6.721.2.3 `std::string decaf::net::URISyntaxException::getInput () const
[inline]`

Returns

the Input string that cause this exception or ""

6.721.2.4 `std::string decaf::net::URISyntaxException::getReason () const` [inline]

Returns

the Reason given for this failure, or ""

The documentation for this class was generated from the following file:

- `src/main/decaf/net/URISyntaxException.h`

6.722 `decaf::internal::net::URIType` Class Reference

Basic type object that holds data that composes a given URI.

```
#include <src/main/decaf/internal/net/URIType.h>
```

Public Member Functions

- **URIType** (const std::string &source)
- **URIType** ()
- virtual ~**URIType** ()
- std::string **getSource** () const
*Gets the source URI string that was parsed to obtain this **URIType** (p. 3126) instance and the resulting data,.*
- void **setSource** (const std::string &source)
*Sets the source URI string that was parsed to obtain this **URIType** (p. 3126) instance and the resulting data,.*
- std::string **getScheme** () const
Gets the Scheme of the URI, e.g.
- void **setScheme** (const std::string &scheme)
Sets the Scheme of the URI, e.g.
- std::string **getSchemeSpecificPart** () const
Gets the Scheme Specific Part of the URI.
- void **setSchemeSpecificPart** (const std::string &schemeSpecificPart)
Sets the Scheme Specific Part of the URI.
- std::string **getAuthority** () const
Gets the Authority of the URI.
- void **setAuthority** (const std::string &authority)
Sets the Authority of the URI.
- std::string **getUserInfo** () const
Gets the user info part of the URI, e.g.

- void **setUserInfo** (const std::string &userinfo)
Sets the user info part of the URI, e.g.
- std::string **getHost** () const
Gets the Host name part of the URI.
- void **setHost** (const std::string &host)
Sets the Host name part of the URI.
- int **getPort** () const
Gets the port part of the URI.
- void **setPort** (int port)
Sets the port part of the URI.
- std::string **getPath** () const
Gets the Path part of the URI.
- void **setPath** (const std::string &path)
Sets the Path part of the URI.
- std::string **getQuery** () const
Gets the Query part of the URI.
- void **setQuery** (const std::string &query)
Sets the Query part of the URI.
- std::string **getFragment** () const
Gets the Fragment part of the URI.
- void **setFragment** (const std::string &fragment)
Sets the Fragment part of the URI.
- bool **isOpaque** () const
Gets if the URI is Opaque.
- void **setOpaque** (bool opaque)
Sets if the URI is Opaque.
- bool **isAbsolute** () const
Gets if the URI is Absolute.
- void **setAbsolute** (bool absolute)
Sets if the URI is Absolute.
- bool **isServerAuthority** () const
Gets if the URI is a Server Authority.
- void **setServerAuthority** (bool serverAuthority)

Sets if the URI is a Server Authority.

- **bool isValid () const**
Gets if the URI is valid, meaning that the source has been set and parsed and all relevant data fields have been set.
- **void setValid (bool valid)**
Sets if the URI is valid, meaning that the source has been set and parsed and all relevant data fields have been set.

6.722.1 Detailed Description

Basic type object that holds data that composes a given URI.

6.722.2 Constructor & Destructor Documentation

6.722.2.1 `decaf::internal::net::URIType::URIType (const std::string & source)`
[inline]

6.722.2.2 `decaf::internal::net::URIType::URIType ()` [inline]

6.722.2.3 `virtual decaf::internal::net::URIType::~~URIType ()` [inline, virtual]

6.722.3 Member Function Documentation

6.722.3.1 `std::string decaf::internal::net::URIType::getAuthority () const`
[inline]

Gets the Authority of the URI.

Returns

Authority part string.

6.722.3.2 `std::string decaf::internal::net::URIType::getFragment () const`
[inline]

Gets the Fragment part of the URI.

Returns

Fragment part string.

6.722.3.3 `std::string decaf::internal::net::URIType::getHost () const` [inline]

Gets the Host name part of the URI.

Returns

Host name part string.

6.722.3.4 `std::string decaf::internal::net::URIType::getPath () const [inline]`

Gets the Path part of the URI.

Returns

Path part string.

6.722.3.5 `int decaf::internal::net::URIType::getPort () const [inline]`

Gets the port part of the URI.

Returns

port part string, -1 if not set.

6.722.3.6 `std::string decaf::internal::net::URIType::getQuery () const [inline]`

Gets the Query part of the URI.

Returns

Query part string.

6.722.3.7 `std::string decaf::internal::net::URIType::getScheme () const [inline]`

Gets the Scheme of the URI, e.g.

scheme ("http"/"ftp"/...).

Returns

scheme part string.

6.722.3.8 `std::string decaf::internal::net::URIType::getSchemeSpecificPart () const [inline]`

Gets the Scheme Specific Part of the URI.

Returns

scheme specific part string.

6.722.3.9 `std::string decaf::internal::net::URIType::getSource () const [inline]`

Gets the source URI string that was parsed to obtain this **URIType** (p.3126) instance and the resulting data,.

Returns

the source URI string

6.722.3.10 `std::string decaf::internal::net::URIType::getUserInfo () const`
[inline]

Gets the user info part of the URI, e.g.

user name, as in `http://user:passwd@host:port/`

Returns

user info part string.

6.722.3.11 `bool decaf::internal::net::URIType::isAbsolute () const` [inline]

Gets if the URI is Absolute.

Returns

true if Absolute.

6.722.3.12 `bool decaf::internal::net::URIType::isOpaque () const` [inline]

Gets if the URI is Opaque.

Returns

true if opaque.

6.722.3.13 `bool decaf::internal::net::URIType::isServerAuthority () const`
[inline]

Gets if the URI is a Server Authority.

Returns

true if Server Authority.

6.722.3.14 `bool decaf::internal::net::URIType::isValid () const` [inline]

Gets if the URI is valid, meaning that the source has been set and parsed and all relevant data fields have been set.

Returns

true if the **URIType** (p. 3126) contains valid data.

6.722.3.15 `void decaf::internal::net::URIType::setAbsolute (bool absolute)`
[inline]

Sets if the URI is Absolute.

Parameters

absolute - true if Absolute.

6.722.3.16 void decaf::internal::net::URIType::setAuthority (const std::string & *authority*) [inline]

Sets the Authority of the URI.

Parameters

authority Authority part string.

6.722.3.17 void decaf::internal::net::URIType::setFragment (const std::string & *fragment*) [inline]

Sets the Fragment part of the URI.

Parameters

fragment - Fragment part string.

6.722.3.18 void decaf::internal::net::URIType::setHost (const std::string & *host*) [inline]

Sets the Host name part of the URI.

Parameters

host - Host name part string.

6.722.3.19 void decaf::internal::net::URIType::setOpaque (bool *opaque*) [inline]

Sets if the URI is Opaque.

Parameters

opaque true if opaque.

6.722.3.20 void decaf::internal::net::URIType::setPath (const std::string & *path*) [inline]

Sets the Path part of the URI.

Parameters

path - Path part string.

6.722.3.21 void decaf::internal::net::URIType::setPort (int *port*) [inline]

Sets the port part of the URI.

Parameters

port - port part string, -1 if not set.

6.722.3.22 `void decaf::internal::net::URIType::setQuery (const std::string & query) [inline]`

Sets the Query part of the URI.

Parameters

query - Query part string.

6.722.3.23 `void decaf::internal::net::URIType::setScheme (const std::string & scheme) [inline]`

Sets the Scheme of the URI, e.g.

scheme ("http"/"ftp"/...).

Parameters

scheme - scheme part string.

6.722.3.24 `void decaf::internal::net::URIType::setSchemeSpecificPart (const std::string & schemeSpecificPart) [inline]`

Sets the Scheme Specific Part of the URI.

Parameters

schemeSpecificPart - scheme specific part string.

6.722.3.25 `void decaf::internal::net::URIType::setServerAuthority (bool serverAuthority) [inline]`

Sets if the URI is a Server Authority.

Parameters

serverAuthority - true if Server Authority.

6.722.3.26 `void decaf::internal::net::URIType::setSource (const std::string & source) [inline]`

Sets the source URI string that was parsed to obtain this **URIType** (p. 3126) instance and the resulting data,.

Parameters

source - the source URI string

6.722.3.27 `void decaf::internal::net::URIType::setUserInfo (const std::string & userinfo) [inline]`

Sets the user info part of the URI, e.g.

user name, as in `http://user:passwd@host:port/`

Parameters

userinfo - user info part string.

6.722.3.28 `void decaf::internal::net::URIType::setValid (bool valid) [inline]`

Sets if the URI is valid, meaning that the source has been set and parsed and all relevant data fields have been set.

Parameters

valid - true if the **URIType** (p. 3126) contains valid data.

The documentation for this class was generated from the following file:

- `src/main/decaf/internal/net/URIType.h`

6.723 decaf::net::URL Class Reference

Class **URL** (p. 3133) represents a Uniform Resource Locator, a pointer to a "resource" on the World Wide Web.

```
#include <src/main/decaf/net/URL.h>
```

Public Member Functions

- **URL** ()
- **URL** (const std::string &url)
- virtual ~**URL** ()

6.723.1 Detailed Description

Class **URL** (p. 3133) represents a Uniform Resource Locator, a pointer to a "resource" on the World Wide Web. A resource can be something as simple as a file or a directory, or it can be a reference to a more complicated object, such as a query to a database or to a search engine. More information on the types of URLs and their formats can be found at:

<http://www.ksc.nasa.gov/facts/internet/url-primer.html>

In general, a **URL** (p. 3133) can be broken into several parts. The previous example of a **URL** (p. 3133) indicates that the protocol to use is http (HyperText Transfer Protocol) and that the information resides on a host machine named `www.ksc.nasa.gov`. The information on that host machine is named `/facts/internet/url-primer.html`. The exact meaning of this name on the host machine is both protocol dependent and host dependent. The information normally resides in a file, but it could be generated on the fly. This component of the **URL** (p. 3133) is called the path component.

A **URL** (p. 3133) can optionally specify a "port", which is the port number to which the TCP connection is made on the remote host machine. If the port is not specified, the default port for the protocol is used instead. For example, the default port for http is 80. An alternative port could be specified as:

```
http://www.ksc.nasa.gov:80/facts/internet/url-primer.html
```

The syntax of **URL** (p. 3133) is defined by RFC 2396: Uniform Resource Identifiers (**URI** (p. 3096)): Generic Syntax, amended by RFC 2732: Format for Literal IPv6 Addresses in URLs. The Literal IPv6 address format also supports scope_ids. The syntax and usage of scope_ids is described here.

A **URL** (p. 3133) may have appended to it a "fragment", also known as a "ref" or a "reference". The fragment is indicated by the sharp sign character "#" followed by more characters. For example,

```
http://www.apache.org/cms/index.html#chapter1
```

This fragment is not technically part of the **URL** (p. 3133). Rather, it indicates that after the specified resource is retrieved, the application is specifically interested in that part of the document that has the tag chapter1 attached to it. The meaning of a tag is resource specific.

An application can also specify a "relative URL", which contains only enough information to reach the resource relative to another **URL** (p. 3133). Relative URLs are frequently used within HTML pages. For example, if the contents of the **URL** (p. 3133):

```
http://www.apache.org/cms/index.html
```

contained within it the relative **URL** (p. 3133):

```
FAQ.html
```

it would be a shorthand for:

```
http://www.apache.org/cms/FAQ.html
```

The relative **URL** (p. 3133) need not specify all the components of a **URL** (p. 3133). If the protocol, host name, or port number is missing, the value is inherited from the fully specified **URL** (p. 3133). The file component must be specified. The optional fragment is not inherited.

The **URL** (p. 3133) class does not itself encode or decode any **URL** (p. 3133) components according to the escaping mechanism defined in RFC2396. It is the responsibility of the caller to encode any fields, which need to be escaped prior to calling **URL** (p. 3133), and also to decode any escaped fields, that are returned from **URL** (p. 3133). Furthermore, because **URL** (p. 3133) has no knowledge of **URL** (p. 3133) escaping, it does not recognise equivalence between the encoded or decoded form of the same **URL** (p. 3133). For example, the two URLs:

```
http://foo.com/hello world/ and http://foo.com/hello%20world
```

would be considered not equal to each other.

Note, the **URI** (p. 3096) class does perform escaping of its component fields in certain circumstances. The recommended way to manage the encoding and decoding of URLs is to use **URI** (p. 3096), and to convert between these two classes using **toURI()** and **URI.toURL()** (p. 3107).

The **URLEncoder** (p. 3136) and **URLDecoder** (p. 3135) classes can also be used, but only for HTML form encoding, which is not the same as the encoding scheme defined in RFC2396.

6.723.2 Constructor & Destructor Documentation

6.723.2.1 decaf::net::URL::URL ()

6.723.2.2 decaf::net::URL::URL (const std::string & *url*)

6.723.2.3 virtual decaf::net::URL::~~URL () [inline, virtual]

The documentation for this class was generated from the following file:

- src/main/decaf/net/URL.h

6.724 decaf::net::URLDecoder Class Reference

```
#include <src/main/decaf/net/URLDecoder.h>
```

Public Member Functions

- virtual ~URLDecoder ()

Static Public Member Functions

- static std::string **decode** (const std::string &value)
Decodes the string argument which is assumed to be encoded in the x-www-form-urlencoded MIME content type.

6.724.1 Constructor & Destructor Documentation

6.724.1.1 virtual decaf::net::URLDecoder::~~URLDecoder () [inline, virtual]

6.724.2 Member Function Documentation

6.724.2.1 static std::string decaf::net::URLDecoder::decode (const std::string & *value*) [static]

Decodes the string argument which is assumed to be encoded in the x-www-form-urlencoded MIME content type.

'+' will be converted to space, " and two following hex digit characters are converted to the equivalent byte value. All other characters are passed through unmodified.

e.g. "A+B+C %24%25" -> "A B C \$%"

Parameters

value - string The encoded string.

Returns

The decoded version as a string.

The documentation for this class was generated from the following file:

- `src/main/decaf/net/URLDecoder.h`

6.725 decaf::net::URLEncoder Class Reference

```
#include <src/main/decaf/net/URLEncoder.h>
```

Public Member Functions

- `virtual ~URLEncoder ()`

Static Public Member Functions

- `static std::string encode (const std::string &value)`

This class contains a utility method for converting a string to the format required by the `application/x-www-form-urlencoded` MIME content type.

6.725.1 Constructor & Destructor Documentation

6.725.1.1 `virtual decaf::net::URLEncoder::~~URLEncoder () [inline, virtual]`

6.725.2 Member Function Documentation

6.725.2.1 `static std::string decaf::net::URLEncoder::encode (const std::string &value) [static]`

This class contains a utility method for converting a string to the format required by the `application/x-www-form-urlencoded` MIME content type.

All characters except letters ('a'..'z', 'A'..'Z') and numbers ('0'..'9') and characters '.', '-', '*', '_' are converted into their hexadecimal value prepended by ".

For example: '#' -> 23

In addition, spaces are substituted by '+'

Parameters

value - the string to be converted

Returns

the converted string

The documentation for this class was generated from the following file:

- `src/main/decaf/net/URLEncoder.h`

6.726 activemq::util::Usage Class Reference

```
#include <src/main/activemq/util/Usage.h>
```

Inheritance diagram for activemq::util::Usage:

Public Member Functions

- virtual `~Usage ()`
- virtual void `waitForSpace ()=0`
*Waits forever for more space to be returned to this **Usage** (p. 3137) Manager.*
- virtual void `waitForSpace (unsigned int timeout)=0`
*Waits for more space to be returned to this **Usage** (p. 3137) Manager, times out when the given time span in milliseconds elapses.*
- virtual void `enqueueUsage (unsigned long long value)=0`
Tries to increase the usage by value amount but blocks if this object is currently full.
- virtual void `increaseUsage (unsigned long long value)=0`
Increases the usage by the value amount.
- virtual void `decreaseUsage (unsigned long long value)=0`
Decreases the usage by the value amount.
- virtual bool `isFull () const =0`
*Returns true if this **Usage** (p. 3137) instance is full, i.e.*

6.726.1 Constructor & Destructor Documentation

6.726.1.1 virtual `activemq::util::Usage::~Usage ()` [inline, virtual]

6.726.2 Member Function Documentation

6.726.2.1 virtual void `activemq::util::Usage::decreaseUsage (unsigned long long value)` [pure virtual]

Decreases the usage by the value amount.

Parameters

value Amount of space to return to the pool

Implemented in `activemq::util::MemoryUsage` (p. 2016).

6.726.2.2 `virtual void activemq::util::Usage::enqueueUsage (unsigned long long value)` [pure virtual]

Tries to increase the usage by *value* amount but blocks if this object is currently full.

Parameters

value Amount of usage in bytes to add.

Implemented in `activemq::util::MemoryUsage` (p. 2016).

6.726.2.3 `virtual void activemq::util::Usage::increaseUsage (unsigned long long value)` [pure virtual]

Increases the usage by the *value* amount.

Parameters

value Amount of usage to add.

Implemented in `activemq::util::MemoryUsage` (p. 2017).

6.726.2.4 `virtual bool activemq::util::Usage::isFull () const` [pure virtual]

Returns true if this `Usage` (p. 3137) instance is full, i.e.

`Usage` (p. 3137) $\geq 100\%$

Returns

true if `Usage` (p. 3137) is at the Full point.

Implemented in `activemq::util::MemoryUsage` (p. 2017).

6.726.2.5 `virtual void activemq::util::Usage::waitForSpace (unsigned int timeout)` [pure virtual]

Waits for more space to be returned to this `Usage` (p. 3137) Manager, times out when the given time span in milliseconds elapses.

Parameters

timeout The time to wait for more space.

Implemented in `activemq::util::MemoryUsage` (p. 2017).

6.726.2.6 `virtual void activemq::util::Usage::waitForSpace ()` [pure virtual]

Waits forever for more space to be returned to this `Usage` (p. 3137) Manager.

Implemented in `activemq::util::MemoryUsage` (p. 2017).

The documentation for this class was generated from the following file:

- `src/main/activemq/util/Usage.h`

6.727 decaf::io::UTFDataFormatException Class Reference

Thrown from classes that attempt to read or write a UTF-8 encoded string and an encoding error is encountered.

```
#include <src/main/decaf/io/UTFDataFormatException.h>
```

Inheritance diagram for decaf::io::UTFDataFormatException:

Public Member Functions

- **UTFDataFormatException** () throw ()
Default Constructor.
- **UTFDataFormatException** (const lang::Exception &ex) throw ()
Copy Constructor.
- **UTFDataFormatException** (const UTFDataFormatException &ex) throw ()
Copy Constructor.
- **UTFDataFormatException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
Constructor - Initializes the file name and line number where this message occurred.
- **UTFDataFormatException** (const std::exception *cause) throw ()
Constructor.
- **UTFDataFormatException** (const char *file, const int lineNumber, const char *msg,...) throw ()
Constructor.
- virtual **UTFDataFormatException** * clone () const
Clones this exception.
- virtual ~**UTFDataFormatException** () throw ()

6.727.1 Detailed Description

Thrown from classes that attempt to read or write a UTF-8 encoded string and an encoding error is encountered.

Since

1.0

6.727.2 Constructor & Destructor Documentation

6.727.2.1 decaf::io::UTFDataFormatException::UTFDataFormatException () throw () [inline]

Default Constructor.

6.727.2.2 `decaf::io::UTFDataFormatException::UTFDataFormatException (const lang::Exception & ex) throw () [inline]`

Copy Constructor.

Parameters

ex the exception to copy

6.727.2.3 `decaf::io::UTFDataFormatException::UTFDataFormatException (const UTFDataFormatException & ex) throw () [inline]`

Copy Constructor.

Parameters

ex the exception to copy, which is an instance of this type

6.727.2.4 `decaf::io::UTFDataFormatException::UTFDataFormatException (const char * file, const int lineNumber, const std::exception * cause, const char * msg, ...) throw () [inline]`

Constructor - Initializes the file name and line number where this message occurred.

Sets the message to report, using an optional list of arguments to parse into the message

Parameters

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.727.2.5 `decaf::io::UTFDataFormatException::UTFDataFormatException (const std::exception * cause) throw () [inline]`

Constructor.

Parameters

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.727.2.6 `decaf::io::UTFDataFormatException::UTFDataFormatException (const char * file, const int lineNumber, const char * msg, ...) throw () [inline]`

Constructor.

Parameters

- file* The file name where exception occurs
- lineNumber* The line number where the exception occurred.
- msg* The message to report
- ... list of primitives that are formatted into the message

6.727.2.7 virtual decaf::io::UTFDataFormatException::~~UTFDataFormatException () throw () [inline, virtual]

6.727.3 Member Function Documentation

6.727.3.1 virtual UTFDataFormatException* decaf::io::UTFDataFormatException::clone () const [inline, virtual]

Clones this exception.

This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns

A new instance of an Exception object that is a copy of this instance.

Reimplemented from **decaf::io::IOException** (p. 1709).

The documentation for this class was generated from the following file:

- src/main/decaf/io/UTFDataFormatException.h

6.728 decaf::util::UUID Class Reference

A class that represents an immutable universally unique identifier (**UUID** (p. 3141)).

```
#include <src/main/decaf/util/UUID.h>
```

Inheritance diagram for decaf::util::UUID:

Public Member Functions

- **UUID** (long long mostSigBits, long long leastSigBits)
*Constructs a new **UUID** (p. 3141) using the specified data.*
- virtual ~**UUID** ()
- virtual int **compareTo** (const **UUID** &value) const
*Compare the given **UUID** (p. 3141) to this one.*
- virtual bool **equals** (const **UUID** &value) const

*Compares this **UUID** (p. 3141) to the one given, returns true if they are equal.*

- virtual bool **operator==** (const **UUID** &value) const
Compares equality between this object and the one passed.
- virtual bool **operator<** (const **UUID** &value) const
Compares this object to another and returns true if this object is considered to be less than the one passed.
- virtual std::string **toString** () const
*Returns a String object representing this **UUID** (p. 3141).*
- virtual long long **getLeastSignificantBits** () const
- virtual long long **getMostSignificantBits** () const
- virtual long long **node** () throw (lang::exceptions::UnsupportedOperationException)
*The node value associated with this **UUID** (p. 3141).*
- virtual long long **timestamp** () throw (lang::exceptions::UnsupportedOperationException)
*The timestamp value associated with this **UUID** (p. 3141).*
- virtual int **clockSequence** () throw (lang::exceptions::UnsupportedOperationException)
*The clock sequence value associated with this **UUID** (p. 3141).*
- virtual int **variant** () throw (lang::exceptions::UnsupportedOperationException)
*The variant number associated with this **UUID** (p. 3141).*
- virtual int **version** () throw (lang::exceptions::UnsupportedOperationException)
*The version number associated with this **UUID** (p. 3141).*

Static Public Member Functions

- static **UUID** **randomUUID** ()
*Static factory to retrieve a type 4 (pseudo randomly generated) **UUID** (p. 3141).*
- static **UUID** **nameUUIDFromBytes** (const std::vector< char > &name)
*Static factory to retrieve a type 3 (name based) **UUID** (p. 3141) based on the specified byte array.*
- static **UUID** **nameUUIDFromBytes** (const char *name, std::size_t size)
*Static factory to retrieve a type 3 (name based) **UUID** (p. 3141) based on the specified byte array.*
- static **UUID** **fromString** (const std::string &name) throw (lang::exceptions::IllegalArgumentException)
*Creates a **UUID** (p. 3141) from the string standard representation as described in the **toString()** (p. 3147) method.*

6.728.1 Detailed Description

A class that represents an immutable universally unique identifier (**UUID** (p. 3141)). A **UUID** (p. 3141) represents a 128-bit value.

There exist different variants of these global identifiers. The methods of this class are for manipulating the Leach-Salz variant, although the constructors allow the creation of any variant of **UUID** (p. 3141) (described below).

The layout of a variant 2 (Leach-Salz) **UUID** (p. 3141) is as follows: The most significant long consists of the following unsigned fields:

```
0xFFFFFFFF00000000 time_low 0x00000000FFFFFF0000 time_mid 0x000000000000F000 version
0x00000000000000FF time_hi
```

The least significant long consists of the following unsigned fields:

```
0xC000000000000000 variant 0x3FFF000000000000 clock_seq 0x0000FFFFFFFFFFFFFF node
```

The variant field contains a value which identifies the layout of the **UUID** (p. 3141). The bit layout described above is valid only for a **UUID** (p. 3141) with a variant value of 2, which indicates the Leach-Salz variant.

The version field holds a value that describes the type of this **UUID** (p. 3141). There are four different basic types of UUIDs: time-based, DCE security, name-based, and randomly generated UUIDs. These types have a version value of 1, 2, 3 and 4, respectively.

For more information including algorithms used to create UUIDs, see the Internet-Draft UUIDs and GUIDs or the standards body definition at ISO/IEC 11578:1996.

6.728.2 Constructor & Destructor Documentation

6.728.2.1 decaf::util::UUID::UUID (long long *mostSigBits*, long long *leastSigBits*)

Constructs a new **UUID** (p. 3141) using the specified data.

mostSigBits is used for the most significant 64 bits of the **UUID** (p. 3141) and *leastSigBits* becomes the least significant 64 bits of the **UUID** (p. 3141).

Parameters

mostSigBits

leastSigBits

6.728.2.2 virtual decaf::util::UUID::~~UUID () [virtual]

6.728.3 Member Function Documentation

6.728.3.1 virtual int decaf::util::UUID::clockSequence () throw (lang::exceptions::UnsupportedOperationException) [virtual]

The clock sequence value associated with this **UUID** (p. 3141).

The 14 bit clock sequence value is constructed from the clock sequence field of this **UUID** (p. 3141). The clock sequence field is used to guarantee temporal uniqueness in a time-based **UUID** (p. 3141).

The clockSequence value is only meaningful in a time-based **UUID** (p. 3141), which has version type 1. If this **UUID** (p. 3141) is not a time-based **UUID** (p. 3141) then this method throws `UnsupportedOperationException`.

Returns

the clockSequence associated with a V1 **UUID** (p. 3141)

Exceptions

`UnsupportedOperationException`

6.728.3.2 `virtual int decaf::util::UUID::compareTo (const UUID & value) const`
[virtual]

Compare the given **UUID** (p. 3141) to this one.

Parameters

value - the **UUID** (p. 3141) to compare to

6.728.3.3 `virtual bool decaf::util::UUID::equals (const UUID & value) const`
[virtual]

Compares this **UUID** (p. 3141) to the one given, returns true if they are equal.

Parameters

value - the **UUID** (p. 3141) to compare to.

Returns

true if UUIDs are the same.

6.728.3.4 `static UUID decaf::util::UUID::fromString (const std::string & name)`
`throw (lang::exceptions::IllegalArgumentException) [static]`

Creates a **UUID** (p. 3141) from the string standard representation as described in the `toString()` (p. 3147) method.

Parameters

name - a string to be used to construct a **UUID** (p. 3141).

Returns

type 3 **UUID** (p. 3141)

6.728.3.5 `virtual long long decaf::util::UUID::getLeastSignificantBits () const`
[virtual]

Returns

the most significant 64 bits of this **UUID**'s 128 bit value.

6.728.3.6 `virtual long long decaf::util::UUID::getMostSignificantBits () const`
[virtual]

Returns

the most significant 64 bits of this UUID's 128 bit value.

6.728.3.7 `static UUID decaf::util::UUID::nameUUIDFromBytes (const`
`std::vector< char > & name)` [static]

Static factory to retrieve a type 3 (name based) **UUID** (p. 3141) based on the specified byte array.

Parameters

name - a byte array to be used to construct a **UUID** (p. 3141).

Returns

type 3 **UUID** (p. 3141)

6.728.3.8 `static UUID decaf::util::UUID::nameUUIDFromBytes (const char *`
`name, std::size_t size)` [static]

Static factory to retrieve a type 3 (name based) **UUID** (p. 3141) based on the specified byte array.

Parameters

name - a byte array to be used to construct a **UUID** (p. 3141).

size - the size of the byte array, or number of bytes to use.

Returns

type 3 **UUID** (p. 3141)

6.728.3.9 `virtual long long decaf::util::UUID::node () throw (`
`lang::exceptions::UnsupportedOperationException)` [virtual]

The node value associated with this **UUID** (p. 3141).

The 48 bit node value is constructed from the node field of this **UUID** (p. 3141). This field is intended to hold the IEEE 802 address of the machine that generated this **UUID** (p. 3141) to guarantee spatial uniqueness.

The node value is only meaningful in a time-based **UUID** (p. 3141), which has version type 1. If this **UUID** (p. 3141) is not a time-based **UUID** (p. 3141) then this method throws `UnsupportedOperationException`.

Returns

the node value of this **UUID** (p. 3141)

Exceptions

UnsupportedOperationException

6.728.3.10 `virtual bool decaf::util::UUID::operator< (const UUID & value)
const [virtual]`

Compares this object to another and returns true if this object is considered to be less than the one passed.

This

Parameters

value - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

6.728.3.11 `virtual bool decaf::util::UUID::operator== (const UUID & value)
const [virtual]`

Compares equality between this object and the one passed.

Parameters

value - the value to be compared to this one.

Returns

true if this object is equal to the one passed.

6.728.3.12 `static UUID decaf::util::UUID::randomUUID () [static]`

Static factory to retrieve a type 4 (pseudo randomly generated) **UUID** (p. 3141).

The **UUID** (p. 3141) is generated using a cryptographically strong pseudo random number generator.

Returns

type 4 **UUID** (p. 3141)

6.728.3.13 `virtual long long decaf::util::UUID::timestamp () throw (
lang::exceptions::UnsupportedOperationException) [virtual]`

The timestamp value associated with this **UUID** (p. 3141).

The 60 bit timestamp value is constructed from the time_low, time_mid, and time_hi fields of this **UUID** (p. 3141). The resulting timestamp is measured in 100-nanosecond units since midnight, October 15, 1582 UTC.

The timestamp value is only meaningful in a time-based **UUID** (p. 3141), which has version type 1. If this **UUID** (p. 3141) is not a time-based **UUID** (p. 3141) then this method throws `UnsupportedOperationException`.

Returns

the timestamp associated with a V1 **UUID** (p. 3141)

Exceptions

UnsupportedOperationException

6.728.3.14 virtual std::string decaf::util::UUID::toString () const [virtual]

Returns a String object representing this **UUID** (p. 3141).

UUID's are formatted as: 00112233-4455-6677-8899-AABBCCDDEEFF whose length is 36.

Returns

formatted string for this **UUID** (p. 3141)

6.728.3.15 virtual int decaf::util::UUID::variant () throw (lang::exceptions::UnsupportedOperationException) [virtual]

The variant number associated with this **UUID** (p. 3141).

The variant number describes the layout of the **UUID** (p. 3141). The variant number has the following meaning:

* 0 Reserved for NCS backward compatibility * 2 The Leach-Salz variant (used by this class) * 6 Reserved, Microsoft Corporation backward compatibility * 7 Reserved for future definition

Returns

the variant associated with a V1 **UUID** (p. 3141)

Exceptions

UnsupportedOperationException

6.728.3.16 virtual int decaf::util::UUID::version () throw (lang::exceptions::UnsupportedOperationException) [virtual]

The version number associated with this **UUID** (p. 3141).

The version number describes how this **UUID** (p. 3141) was generated. The version number has the following meaning:

* 1 Time-based **UUID** (p. 3141) * 2 DCE security **UUID** (p. 3141) * 3 Name-based **UUID** (p. 3141) * 4 Randomly generated **UUID** (p. 3141)

Returns

the version associated with a V1 **UUID** (p. 3141)

Exceptions

UnsupportedOperationException

The documentation for this class was generated from the following file:

- src/main/decaf/util/**UUID.h**

6.729 activemq::wireformat::WireFormat Class Reference

Provides a mechanism to marshal commands into and out of packets or into and out of streams, Channels and Datagrams.

```
#include <src/main/activemq/wireformat/WireFormat.h>
```

Inheritance diagram for activemq::wireformat::WireFormat:

Public Member Functions

- virtual `~WireFormat ()`
- virtual void **marshal** (const **Pointer**< **commands::Command** > &command, const **activemq::transport::Transport** *transport, **decaf::io::DataOutputStream** *out)=0
throw (**decaf::io::IOException**)
Stream based marshaling of a Command, this method blocks until the entire Command has been written out to the output stream.
- virtual **Pointer**< **commands::Command** > **unmarshal** (const **activemq::transport::Transport** *transport, **decaf::io::DataInputStream** *in)=0
throw (**decaf::io::IOException**)
Stream based unmarshaling, blocks on reads on the input stream until a complete command has been read and unmarshaled into the correct form.
- virtual void **setVersion** (int version)=0
Set the Version.
- virtual int **getVersion** () const =0
Get the Version.
- virtual bool **hasNegotiator** () const =0
*Returns true if this **WireFormat** (p. 3148) has a Negotiator that needs to wrap the Transport that uses it.*
- virtual bool **inReceive** () const =0
Indicates if the WireFormat object is in the process of receiving a message.
- virtual **Pointer**< **transport::Transport** > **createNegotiator** (const **Pointer**< **transport::Transport** > &transport)=0 throw (**decaf::lang::exceptions::UnsupportedOperationException**)
If the Transport Provides a Negotiator this method will create and return a new instance of the Negotiator.

6.729.1 Detailed Description

Provides a mechanism to marshal commands into and out of packets or into and out of streams, Channels and Datagrams.

Version

Revision:

1.1

6.729.2 Constructor & Destructor Documentation

6.729.2.1 virtual `activemq::wireformat::WireFormat::~WireFormat ()` [inline, virtual]

6.729.3 Member Function Documentation

6.729.3.1 virtual `Pointer<transport::Transport> activemq::wireformat::WireFormat::createNegotiator (const Pointer< transport::Transport > & transport) throw (decaf::lang::exceptions::UnsupportedOperationException)` [pure virtual]

If the Transport Provides a Negotiator this method will create and return a new instance of the Negotiator.

Parameters

transport - the Transport to Wrap the Negotiator around.

Returns

new instance of a **WireFormatNegotiator** (p. 3182) as a **Pointer<Transport>** (p. 2343).

Exceptions

UnsupportedOperationException if the **WireFormat** (p. 3148) doesn't have a Negotiator.

Implemented in `activemq::wireformat::openwire::OpenWireFormat` (p. 2300), and `activemq::wireformat::stomp::StompWireFormat` (p. 2884).

6.729.3.2 virtual `int activemq::wireformat::WireFormat::getVersion () const` [pure virtual]

Get the Version.

Returns

the version of the wire format

Implemented in `activemq::wireformat::openwire::OpenWireFormat` (p. 2302), and `activemq::wireformat::stomp::StompWireFormat` (p. 2884).

6.729.3.3 `virtual bool activemq::wireformat::WireFormat::hasNegotiator () const` [pure virtual]

Returns true if this **WireFormat** (p. 3148) has a Negotiator that needs to wrap the Transport that uses it.

Returns

true if the **WireFormat** (p. 3148) provides a Negotiator.

Implemented in **activemq::wireformat::openwire::OpenWireFormat** (p. 2302), and **activemq::wireformat::stomp::StompWireFormat** (p. 2885).

6.729.3.4 `virtual bool activemq::wireformat::WireFormat::inReceive () const` [pure virtual]

Indicates if the WireFormat object is in the process of receiving a message.

This is useful for monitoring inactivity and the **WireFormat** (p. 3148) is processing a large message which takes longer than some configured timeout to unmarshal, the inactivity monitor can query the **WireFormat** (p. 3148) instance to determine if its busy or not and not mark the connection as inactive if so.

Returns

true if the **WireFormat** (p. 3148) object is unmarshaling a message.

Implemented in **activemq::wireformat::openwire::OpenWireFormat** (p. 2302), and **activemq::wireformat::stomp::StompWireFormat** (p. 2885).

6.729.3.5 `virtual void activemq::wireformat::WireFormat::marshal (const Pointer< commands::Command > & command, const activemq::transport::Transport * transport, decaf::io::DataOutputStream * out) throw (decaf::io::IOException)` [pure virtual]

Stream based marshaling of a Command, this method blocks until the entire Command has been written out to the output stream.

Parameters

command The Command to Marshal

transport The Transport that called this method.

out The output stream to write the command to.

Exceptions

IOException

Implemented in **activemq::wireformat::openwire::OpenWireFormat** (p. 2304), and **activemq::wireformat::stomp::StompWireFormat** (p. 2885).

6.729.3.6 virtual void activemq::wireformat::WireFormat::setVersion (int *version*) [pure virtual]

Set the Version.

Parameters

version the version of the wire format

Implemented in **activemq::wireformat::openwire::OpenWireFormat** (p. 2306), and **activemq::wireformat::stomp::StompWireFormat** (p. 2885).

6.729.3.7 virtual Pointer<commands::Command> activemq::wireformat::WireFormat::unmarshal (const activemq::transport::Transport * *transport*, decaf::io::DataInputStream * *in*) throw (decaf::io::IOException) [pure virtual]

Stream based unmarshaling, blocks on reads on the input stream until a complete command has been read and unmarshaled into the correct form.

Returns a Pointer to the newly unmarshaled Command.

Parameters

transport - Pointer to the transport that is making this request.

in - the input stream to read the command from.

Returns

the newly marshaled Command, caller owns the pointer

Exceptions

IOException

Implemented in **activemq::wireformat::openwire::OpenWireFormat** (p. 2308), and **activemq::wireformat::stomp::StompWireFormat** (p. 2886).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/**WireFormat.h**

6.730 activemq::wireformat::WireFormatFactory Class Reference

The **WireFormatFactory** (p. 3151) is the interface that all **WireFormatFactory** (p. 3151) classes must extend.

```
#include <src/main/activemq/wireformat/WireFormatFactory.h>
```

Inheritance diagram for activemq::wireformat::WireFormatFactory:

Public Member Functions

- virtual `~WireFormatFactory()`
- virtual `Pointer<WireFormat> createWireFormat (const decaf::util::Properties &properties)=0` throw (decaf::lang::exceptions::IllegalStateException)
*Creates a new **WireFormat** (p. 3148) Object passing it a set of properties from which it can obtain any optional settings.*

6.730.1 Detailed Description

The **WireFormatFactory** (p. 3151) is the interface that all **WireFormatFactory** (p. 3151) classes must extend. The Factory creates a **WireFormat** (p. 3148) Object based on the properties that are set in the passed **Properties** object.

6.730.2 Constructor & Destructor Documentation

- 6.730.2.1** `virtual activemq::wireformat::WireFormatFactory::~~WireFormatFactory ()` [inline, virtual]

6.730.3 Member Function Documentation

- 6.730.3.1** `virtual Pointer<WireFormat> activemq::wireformat::WireFormatFactory::createWireFormat (const decaf::util::Properties & properties)` throw (decaf::lang::exceptions::IllegalStateException) [pure virtual]

Creates a new **WireFormat** (p. 3148) Object passing it a set of properties from which it can obtain any optional settings.

Parameters

properties - the Properties for this **WireFormat** (p. 3148)

Returns

Pointer to a new instance of a **WireFormat** (p. 3148) object.

Implemented in **activemq::wireformat::openwire::OpenWireFormatFactory** (p. 2309), and **activemq::wireformat::stomp::StompWireFormatFactory** (p. 2887).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/WireFormatFactory.h`

6.731 activemq::commands::WireFormatInfo Class Reference

```
#include <src/main/activemq/commands/WireFormatInfo.h>
```

Inheritance diagram for `activemq::commands::WireFormatInfo`:

Public Member Functions

- **WireFormatInfo** ()
- virtual **~WireFormatInfo** ()
- virtual unsigned char **getDataStructureType** () const
Get the unique identifier that this object and its own Marshaler share.
- virtual **DataStructure** * **cloneDataStructure** () const
Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.
- virtual void **copyDataStructure** (const **DataStructure** *src)
Copy the contents of the passed object into this objects members, overwriting any existing data.
- virtual std::string **toString** () const
*Returns a string containing the information for this **DataStructure** (p. 1372) such as its type and value of its elements.*
- virtual bool **equals** (const **DataStructure** *value) const
*Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent.*
- virtual bool **isMarshalAware** () const
Indicates that this command is aware of Marshaling, and needs to have its Marshaling methods invoked.
- virtual **decaf::lang::Pointer**< **commands::Command** > **visit** (**activemq::state::CommandVisitor** *visitor) throw (**exceptions::ActiveMQException**)
Allows a Visitor to visit this command and return a response to the command based on the command type being visited.
- int **getVersion** () const
Get the current Wireformat Version.
- void **setVersion** (int version)
Set the current Wireformat Version.
- long long **getMaxInactivityDuration** () const
Returns the currently configured Max Inactivity duration.
- void **setMaxInactivityDuration** (long long maxInactivityDuration)
Sets the Max inactivity duration value.
- long long **getMaxInactivityDurationInitialDelay** () const
Returns the currently configured Max Inactivity Initial Delay duration.
- void **setMaxInactivityDurationInitialDelay** (long long maxInactivityDurationInitialDelay)
Sets the Max inactivity initial delay duration value.
- bool **isStackTraceEnabled** () const

Checks if the stackTraceEnabled flag is on.

- void **setStackTraceEnabled** (bool stackTraceEnabled)
Sets if the stackTraceEnabled flag is on.
- bool **isTcpNoDelayEnabled** () const
Checks if the tcpNoDelayEnabled flag is on.
- void **setTcpNoDelayEnabled** (bool tcpNoDelayEnabled)
Sets if the tcpNoDelayEnabled flag is on.
- bool **isCacheEnabled** () const
Checks if the cacheEnabled flag is on.
- void **setCacheEnabled** (bool cacheEnabled)
Sets if the cacheEnabled flag is on.
- int **getCacheSize** () const
Gets the Cache Size setting.
- void **setCacheSize** (int value)
Sets the Cache Size setting.
- bool **isTightEncodingEnabled** () const
Checks if the tightEncodingEnabled flag is on.
- void **setTightEncodingEnabled** (bool tightEncodingEnabled)
Sets if the tightEncodingEnabled flag is on.
- bool **isSizePrefixDisabled** () const
Checks if the sizePrefixDisabled flag is on.
- void **setSizePrefixDisabled** (bool sizePrefixDisabled)
Sets if the sizePrefixDisabled flag is on.
- const std::vector< unsigned char > & **getMagic** () const
Get the Magic field.
- void **setMagic** (const std::vector< unsigned char > &magic)
Sets the value of the magic field.
- const std::vector< unsigned char > & **getMarshalledProperties** () const
Get the marshalledProperties field.
- void **setMarshalledProperties** (const std::vector< unsigned char > &marshalledProperties)
Sets the value of the marshalledProperties field.
- virtual const **util::PrimitiveMap** & **getProperties** () const
*Gets the Properties for this **Command** (p. 991).*

- virtual **util::PrimitiveMap** & **getProperties** ()
*Gets the Properties for this **Command** (p. 991).*
- virtual void **setProperties** (const **util::PrimitiveMap** &map)
*Sets the Properties for this **Command** (p. 991).*
- bool **isValid** () const
*Determines if we think this is a Valid **WireFormatInfo** (p. 3152) command.*
- virtual bool **isWireFormatInfo** () const
- virtual void **beforeMarshal** (**wireformat::WireFormat** *wireFormat **AMQCPP_** - **UNUSED**) throw (**decaf::io::IOException**)
Handles the marshaling of the objects properties into the internal byte array before the object is marshalled to the wire.
- virtual void **afterUnmarshal** (**wireformat::WireFormat** *wireFormat **AMQCPP_** - **UNUSED**) throw (**decaf::io::IOException**)
Called after unmarshaling is started to cleanup the object being unmarshaled.

Static Public Attributes

- static const unsigned char **ID_WIREFORMATINFO** = 1

6.731.1 Constructor & Destructor Documentation

- 6.731.1.1 **activemq::commands::WireFormatInfo::WireFormatInfo** ()
- 6.731.1.2 **virtual activemq::commands::WireFormatInfo::~~WireFormatInfo** ()
[virtual]

6.731.2 Member Function Documentation

- 6.731.2.1 **virtual void activemq::commands::WireFormatInfo::afterUnmarshal** (**wireformat::WireFormat** *wireFormat **AMQCPP_** **UNUSED**) throw (**decaf::io::IOException**) [virtual]

Called after unmarshaling is started to cleanup the object being unmarshaled.

Parameters

wireFormat - the wireformat object to control unmarshaling

Reimplemented from **activemq::commands::BaseDataStructure** (p. 660).

- 6.731.2.2 **virtual void activemq::commands::WireFormatInfo::beforeMarshal** (**wireformat::WireFormat** *wireFormat **AMQCPP_** **UNUSED**) throw (**decaf::io::IOException**) [virtual]

Handles the marshaling of the objects properties into the internal byte array before the object is marshalled to the wire.

Parameters

wireFormat - the wire formatting controller

Reimplemented from `activemq::commands::BaseDataStructure` (p. 660).

6.731.2.3 `virtual DataStructure* activemq::commands::WireFormatInfo::cloneDataStructure () const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

Implements `activemq::commands::DataStructure` (p. 1373).

6.731.2.4 `virtual void activemq::commands::WireFormatInfo::copyDataStructure (const DataStructure * src) [virtual]`

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns

src - Source Object

Reimplemented from `activemq::commands::BaseCommand` (p. 598).

6.731.2.5 `virtual bool activemq::commands::WireFormatInfo::equals (const DataStructure * value) const [virtual]`

Compares the `DataStructure` (p. 1372) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

Reimplemented from `activemq::commands::BaseCommand` (p. 598).

6.731.2.6 `int activemq::commands::WireFormatInfo::getCacheSize () const`

Gets the Cache Size setting.

Returns

currently set cache size.

6.731.2.7 `virtual unsigned char activemq::commands::WireFormatInfo::getDataStructureType () const`
[virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataStructure** (p. 1372) type copy.

Implements **activemq::commands::DataStructure** (p. 1375).

6.731.2.8 `const std::vector<unsigned char>& activemq::commands::WireFormatInfo::getMagic () const`
[inline]

Get the Magic field.

Returns

const reference to a `std::vector<char>`

6.731.2.9 `const std::vector<unsigned char>& activemq::commands::WireFormatInfo::getMarshaledProperties () const`
[inline]

Get the marshalledProperties field.

Returns

const reference to a `std::vector<char>`

6.731.2.10 `long long activemq::commands::WireFormatInfo::getMaxInactivityDuration () const`

Returns the currently configured Max Inactivity duration.

Returns

the set inactivity duration value.

6.731.2.11 `long long activemq::commands::WireFormatInfo::getMaxInactivityDurationInitalDelay () const`

Returns the currently configured Max Inactivity Intial Delay duration.

Returns

the set inactivity duration initial delay value.

6.731.2.12 `virtual util::PrimitiveMap& activemq::commands::WireFormatInfo::getProperties ()`
[inline, virtual]

Gets the Properties for this **Command** (p. 991).

Returns

the Properties object for this **Command** (p. 991).

6.731.2.13 `virtual const util::PrimitiveMap& activemq::commands::WireFormatInfo::getProperties ()`
`const` [inline, virtual]

Gets the Properties for this **Command** (p. 991).

Returns

the Properties object for this **Command** (p. 991).

6.731.2.14 `int activemq::commands::WireFormatInfo::getVersion () const`
[inline]

Get the current Wireformat Version.

Returns

int that identifies the version

6.731.2.15 `bool activemq::commands::WireFormatInfo::isCacheEnabled () const`

Checks if the cacheEnabled flag is on.

Returns

true if the flag is on.

6.731.2.16 `virtual bool activemq::commands::WireFormatInfo::isMarshalAware () const` [inline, virtual]

Indicates that this command is aware of Marshaling, and needs to have its Marshaling methods invoked.

Returns

boolean indicating desire to be in marshaling stages

Reimplemented from **activemq::commands::BaseDataStructure** (p. 662).

6.731.2.17 `bool activemq::commands::WireFormatInfo::isSizePrefixDisabled () const`

Checks if the sizePrefixDisabled flag is on.

Returns

true if the flag is on.

6.731.2.18 `bool activemq::commands::WireFormatInfo::isStackTraceEnabled () const`

Checks if the stackTraceEnabled flag is on.

Returns

true if the flag is on.

6.731.2.19 `bool activemq::commands::WireFormatInfo::isTcpNoDelayEnabled () const`

Checks if the tcpNoDelayEnabled flag is on.

Returns

true if the flag is on.

6.731.2.20 `bool activemq::commands::WireFormatInfo::isTightEncodingEnabled () const`

Checks if the tightEncodingEnabled flag is on.

Returns

true if the flag is on.

6.731.2.21 `bool activemq::commands::WireFormatInfo::isValid () const`

Determines if we think this is a Valid **WireFormatInfo** (p. 3152) command.

Returns

true if its valid.

6.731.2.22 `virtual bool activemq::commands::WireFormatInfo::isWireFormatInfo () const [inline, virtual]`

Returns

answers true to the isWireFormatInfo query

Reimplemented from **activemq::commands::BaseCommand** (p. 602).

6.731.2.23 `void activemq::commands::WireFormatInfo::setCacheEnabled (bool cacheEnabled)`

Sets if the `cacheEnabled` flag is on.

Parameters

cacheEnabled - true to turn flag is on

6.731.2.24 `void activemq::commands::WireFormatInfo::setCacheSize (int value)`

Sets the Cache Size setting.

Parameters

value - value to set to the cache size.

6.731.2.25 `void activemq::commands::WireFormatInfo::setMagic (const std::vector< unsigned char > & magic) [inline]`

Sets the value of the magic field.

Parameters

magic - const std::vector<char>

6.731.2.26 `void activemq::commands::WireFormatInfo::setMarshaledProperties (const std::vector< unsigned char > & marshalledProperties) [inline]`

Sets the value of the marshalledProperties field.

Parameters

marshalledProperties The Byte Array vector that contains the marshaled form of the Message (p. 2018) properties, this is the data sent over the wire.

6.731.2.27 `void activemq::commands::WireFormatInfo::setMaxInactivityDuration (long long maxInactivityDuration)`

Sets the Max inactivity duration value.

Parameters

maxInactivityDuration - max time a client can be inactive.

6.731.2.28 `void activemq::commands::WireFormatInfo::setMaxInactivityDurationInitialDelay (long long maxInactivityDurationInitialDelay)`

Sets the Max inactivity initial delay duration value.

Parameters

maxInactivityDurationInitialDelay - time before the inactivity delay is checked.

6.731.2.29 `virtual void activemq::commands::WireFormatInfo::setProperties (const util::PrimitiveMap & map) [inline, virtual]`

Sets the Properties for this **Command** (p. 991).

Parameters

map - PrimitiveMap to copy

6.731.2.30 `void activemq::commands::WireFormatInfo::setSizePrefixDisabled (bool sizePrefixDisabled)`

Sets if the sizePrefixDisabled flag is on.

Parameters

sizePrefixDisabled - true to turn flag is on

6.731.2.31 `void activemq::commands::WireFormatInfo::setStackTraceEnabled (bool stackTraceEnabled)`

Sets if the stackTraceEnabled flag is on.

Parameters

stackTraceEnabled - ture to turn flag is on

6.731.2.32 `void activemq::commands::WireFormatInfo::setTcpNoDelayEnabled (bool tcpNoDelayEnabled)`

Sets if the tcpNoDelayEnabled flag is on.

Parameters

tcpNoDelayEnabled - ture to turn flag is on

6.731.2.33 `void activemq::commands::WireFormatInfo::setTightEncodingEnabled (bool tightEncodingEnabled)`

Sets if the `tightEncodingEnabled` flag is on.

Parameters

tightEncodingEnabled - true to turn flag is on

6.731.2.34 `void activemq::commands::WireFormatInfo::setVersion (int version) [inline]`

Set the current Wireformat Version.

Parameters

version - int that identifies the version

6.731.2.35 `virtual std::string activemq::commands::WireFormatInfo::toString () const [virtual]`

Returns a string containing the information for this **DataStructure** (p.1372) such as its type and value of its elements.

Returns

formatted string useful for debugging.

Reimplemented from `activemq::commands::BaseCommand` (p.602).

6.731.2.36 `virtual decaf::lang::Pointer<commands::Command> activemq::commands::WireFormatInfo::visit (activemq::state::CommandVisitor * visitor) throw (exceptions::ActiveMQException) [virtual]`

Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

The command will call the proper `processXXX` method in the visitor.

Returns

a **Response** (p.2611) to the visitor being called or NULL if no response.

Implements `activemq::commands::Command` (p.995).

6.731.3 Field Documentation

6.731.3.1 `const unsigned char activemq::commands::WireFormatInfo::ID_WIREFORMATINFO = 1 [static]`

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/WireFormatInfo.h`

6.732 activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller Class Reference

Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3163).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/WireFormatInfoMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller:

Public Member Functions

- **WireFormatInfoMarshaller** ()
- virtual **~WireFormatInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.732.1 Detailed Description

Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3163). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.732.2 Constructor & Destructor Documentation

6.732.2.1 `activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller::WireFormatInfoMarshaller()` [inline]

6.732.2.2 `virtual activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller::~~WireFormatInfoMarshaller()` [inline, virtual]

6.732.3 Member Function Documentation

6.732.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller::createObject() const` [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.732.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller::getDataStructureType() const` [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.732.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller::looseMarshal(OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1342).

6.732.3.4 virtual void `activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller::looseUnmarshal` (`OpenWireFormat * wireFormat`, `commands::DataStructure * dataStructure`, `decaf::io::DataInputStream * dataIn`) throw (`decaf::io::IOException`) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1348).

6.732.3.5 virtual int `activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller::tightMarshal` (`OpenWireFormat * wireFormat`, `commands::DataStructure * dataStructure`, `utils::BooleanStream * bs`) throw (`decaf::io::IOException`) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1354).

6.732.3.6 virtual void activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1360).

6.732.3.7 virtual void activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.
bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1366).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v2/WireFormatInfoMarshaller.h`

6.733 activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller Class Reference

Marshaling code for Open Wire Format for `WireFormatInfoMarshaller` (p. 3166).

#include <src/main/activemq/wireformat/openwire/marshal/v5/WireFormatInfoMarshaller.h>

Inheritance diagram for activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller:

Public Member Functions

- **WireFormatInfoMarshaller** ()
- virtual **~WireFormatInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.733.1 Detailed Description

Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3166). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.733.2 Constructor & Destructor Documentation

6.733.2.1 `activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller::WireFormatInfoMarshaller () [inline]`

6.733.2.2 `virtual activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller::~~WireFormatInfoMarshaller () [inline, virtual]`

6.733.3 Member Function Documentation

6.733.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.733.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.733.3.3 `virtual void activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1342).

6.733.3.4 virtual void activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1348).

6.733.3.5 virtual int activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1354).

6.733.3.6 virtual void activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1360).

```
6.733.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1366).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/WireFormatInfoMarshaller.h`

6.734 **activemq::wireformat::openwire::marshal::v1::WireFormatInfoMar** **Class Reference**

Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3170).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/WireFormatInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller**:

Public Member Functions

- **WireFormatInfoMarshaller** ()
- virtual \sim **WireFormatInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (**decaf::io::IOException**)
Write a object instance to data output stream.

6.734.1 Detailed Description

Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3170). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.734.2 Constructor & Destructor Documentation

6.734.2.1 `activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller::WireFormatInfoMarshaller () [inline]`

6.734.2.2 `virtual activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller::~~WireFormatInfoMarshaller () [inline, virtual]`

6.734.3 Member Function Documentation

6.734.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.734.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.734.3.3 `virtual void activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1342).

6.734.3.4 virtual void activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1348).

6.734.3.5 virtual int activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1354).

6.734.3.6 virtual void activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1360).

```
6.734.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1366).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/WireFormatInfoMarshaller.h`

6.735 **activemq::wireformat::openwire::marshal::v4::WireFormatInfoMar** **Class Reference**

Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3174).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/WireFormatInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller**:

Public Member Functions

- **WireFormatInfoMarshaller** ()
- virtual **~WireFormatInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.735.1 Detailed Description

Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3174). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.735.2 Constructor & Destructor Documentation

6.735.2.1 `activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller::WireFormatInfoMarshaller () [inline]`

6.735.2.2 `virtual activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller::~~WireFormatInfoMarshaller () [inline, virtual]`

6.735.3 Member Function Documentation

6.735.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.735.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.735.3.3 `virtual void activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1342).

6.735.3.4 virtual void activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1348).

6.735.3.5 virtual int activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1354).

6.735.3.6 virtual void activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1360).

```
6.735.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1366).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/WireFormatInfoMarshaller.h`

6.736 **activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller** Class Reference

Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3178).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/WireFormatInfoMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller**:

Public Member Functions

- **WireFormatInfoMarshaller** ()
- virtual **~WireFormatInfoMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaller.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual int **tightMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)
Un-marshall an object instance from the data input stream.
- virtual void **looseMarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)
Write a object instance to data output stream.

6.736.1 Detailed Description

Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3178). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.736.2 Constructor & Destructor Documentation

6.736.2.1 `activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller::WireFormatInfoMarshaller () [inline]`

6.736.2.2 `virtual activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller::~~WireFormatInfoMarshaller () [inline, virtual]`

6.736.3 Member Function Documentation

6.736.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1331).

6.736.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.736.3.3 `virtual void activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException) [virtual]`

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1342).

6.736.3.4 virtual void activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1348).

6.736.3.5 virtual int activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1354).

6.736.3.6 virtual void activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1360).

```
6.736.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1366).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**WireFormatInfoMarshaller.h**

6.737 activemq::wireformat::WireFormatNegotiator Class Reference

Defines a **WireFormatNegotiator** (p. 3182) which allows a **WireFormat** (p. 3148) to.

```
#include <src/main/activemq/wireformat/WireFormatNegotiator.h>
```

Inheritance diagram for **activemq::wireformat::WireFormatNegotiator**:

Public Member Functions

- **WireFormatNegotiator** (const **Pointer**< **transport::Transport** > &next)
Constructor.
- virtual ~**WireFormatNegotiator** ()

6.737.1 Detailed Description

Defines a **WireFormatNegotiator** (p. 3182) which allows a **WireFormat** (p. 3148) to.

6.737.2 Constructor & Destructor Documentation

- 6.737.2.1** **activemq::wireformat::WireFormatNegotiator::WireFormatNegotiator** (
const **Pointer**< **transport::Transport** > & *next*) [inline]

Constructor.

Parameters

next - the next Transport in the chain

- 6.737.2.2** virtual
activemq::wireformat::WireFormatNegotiator::~~WireFormatNegotiator (
) [inline, virtual]

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/**WireFormatNegotiator.h**

6.738 activemq::wireformat::WireFormatRegistry Class Reference

Registry of all **WireFormat** (p. 3148) Factories that are available to the client at runtime.

```
#include <src/main/activemq/wireformat/WireFormatRegistry.h>
```

Public Member Functions

- virtual ~**WireFormatRegistry** ()
- **WireFormatFactory** * **findFactory** (const std::string &name) const throw (decaf::lang::exceptions::NoSuchElementException)
*Gets a Registered **WireFormatFactory** (p. 3151) from the Registry and returns it if there is not a registered format factory with the given name an exception is thrown.*
- void **registerFactory** (const std::string &name, **WireFormatFactory** *factory) throw (decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::NullPointerException)

*Registers a new **WireFormatFactory** (p. 3151) with this Registry.*

- void **unregisterFactory** (const std::string &name)
Unregisters the Factory with the given name and deletes that instance of the Factory.
- std::vector< std::string > **getWireFormatNames** () const
Retrieves a list of the names of all the Registered WireFormat's in this Registry.

Static Public Member Functions

- static **WireFormatRegistry** & **getInstance** ()
*Gets the single instance of the **WireFormatRegistry** (p. 3183).*

6.738.1 Detailed Description

Registry of all **WireFormat** (p. 3148) Factories that are available to the client at runtime. New WireFormat's must have a factory registered here before a connection attempt is made.

Since

3.0

6.738.2 Constructor & Destructor Documentation

- 6.738.2.1** virtual
activemq::wireformat::WireFormatRegistry::~~WireFormatRegistry ()
[virtual]

6.738.3 Member Function Documentation

- 6.738.3.1** WireFormatFactory* ac-
tivemq::wireformat::WireFormatRegistry::findFactory
(const std::string & *name*) const throw (
decaf::lang::exceptions::NoSuchElementException)

Gets a Registered **WireFormatFactory** (p. 3151) from the Registry and returns it if there is not a registered format factory with the given name an exception is thrown.

Parameters

name The name of the Factory to find in the Registry.

Returns

the Factory registered under the given name.

Exceptions

NoSuchElementException if no factory is registered with that name.

6.738.3.2 static WireFormatRegistry& activemq::wireformat::WireFormatRegistry::getInstance () [static]

Gets the single instance of the **WireFormatRegistry** (p. 3183).

Returns

reference to the single instance of this Registry

6.738.3.3 std::vector<std::string> activemq::wireformat::WireFormatRegistry::getWireFormatNames () const

Retrieves a list of the names of all the Registered WireFormat's in this Registry.

Returns

stl vector of strings with all the **WireFormat** (p. 3148) names registered.

6.738.3.4 void activemq::wireformat::WireFormatRegistry::registerFactory (const std::string & *name*, WireFormatFactory * *factory*) throw (decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::NullPointerException)

Registers a new **WireFormatFactory** (p. 3151) with this Registry.

If a Factory with the given name is already registered it is overwritten with the new one. Once a factory is added to the Registry its lifetime is controlled by the Registry, it will be deleted once the Registry has been deleted.

Parameters

name The name of the new Factory to register.

factory The new Factory to add to the Registry.

Exceptions

IllegalArgumentException is name is the empty string.

NullPointerException if the Factory is Null.

6.738.3.5 void activemq::wireformat::WireFormatRegistry::unregisterFactory (const std::string & *name*)

Unregisters the Factory with the given name and deletes that instance of the Factory.

Parameters

name Name of the Factory to unregister and destroy

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/**WireFormatRegistry.h**

6.739 activemq::transport::inactivity::WriteChecker Class Reference

Runnable class used by the {}.

```
#include <src/main/activemq/transport/inactivity/WriteChecker.h>
```

Inheritance diagram for `activemq::transport::inactivity::WriteChecker`:

Public Member Functions

- **WriteChecker** (**InactivityMonitor** *parent)
- virtual ~**WriteChecker** ()
- virtual void **run** ()

Run method - called by the Thread class in the context of the thread.

6.739.1 Detailed Description

Runnable class used by the {}.

See also

InactivityMonitor (p. 1624)} to make periodic writes to the underlying **transport** (p. 74) if no other write activity is going on in order to more quickly detect failures of the connection to the broker.

Since

3.1.0

6.739.2 Constructor & Destructor Documentation

6.739.2.1 `activemq::transport::inactivity::WriteChecker::WriteChecker (InactivityMonitor * parent)`

6.739.2.2 `virtual activemq::transport::inactivity::WriteChecker::~~WriteChecker ()` [virtual]

6.739.3 Member Function Documentation

6.739.3.1 `virtual void activemq::transport::inactivity::WriteChecker::run ()` [virtual]

Run method - called by the Thread class in the context of the thread.

Implements **decaf::lang::Runnable** (p. 2642).

The documentation for this class was generated from the following file:

- `src/main/activemq/transport/inactivity/WriteChecker.h`

6.740 decaf::io::Writer Class Reference

```
#include <src/main/decaf/io/Writer.h>
```

Public Member Functions

- virtual `~Writer ()`
- virtual void `setOutputStream (OutputStream *os)=0`
Sets the target output stream.
- virtual `OutputStream * getOutputStream ()=0`
Gets the target output stream.
- virtual void `write (const unsigned char *buffer, std::size_t count)=0` throw (`IOException`, `lang::exceptions::NullPointerException`)
Writes a byte array to the output stream.
- virtual void `writeByte (unsigned char v)=0` throw (`IOException`)
Writes a byte to the output stream.

6.740.1 Constructor & Destructor Documentation

6.740.1.1 virtual `decaf::io::Writer::~~Writer ()` [inline, virtual]

6.740.2 Member Function Documentation

6.740.2.1 virtual `OutputStream* decaf::io::Writer::getOutputStream ()` [pure virtual]

Gets the target output stream.

Returns

the output stream currently being used

6.740.2.2 virtual void `decaf::io::Writer::setOutputStream (OutputStream * os)` [pure virtual]

Sets the target output stream.

Parameters

os The provided Outputstream to use to write to.

6.740.2.3 `virtual void decaf::io::Writer::write (const unsigned char
* buffer, std::size_t count) throw (IOException,
lang::exceptions::NullPointerException) [pure virtual]`

Writes a byte array to the output stream.

Parameters

- buffer* a byte array
- count* the number of bytes in the array to write.

Exceptions

IOException (p. 1707) thrown if an error occurs.

6.740.2.4 `virtual void decaf::io::Writer::writeByte (unsigned char v) throw (IOException) [pure virtual]`

Writes a byte to the output stream.

Parameters

- v* The value to be written.

Exceptions

IOException (p. 1707) thrown if an error occurs.

The documentation for this class was generated from the following file:

- `src/main/decaf/io/Writer.h`

6.741 decaf::security::auth::x500::X500Principal Class Reference

`#include <src/main/decaf/security/auth/x500/X500Principal.h>`

Inheritance diagram for decaf::security::auth::x500::X500Principal:

Public Member Functions

- virtual `~X500Principal ()`
- virtual `std::string getName () const =0`
Provides the name of this principal.
- virtual void `getEncoded (std::vector< unsigned char > &output) const =0`
- virtual int `hashCode () const =0`

6.741.1 Constructor & Destructor Documentation

6.741.1.1 virtual decaf::security::auth::x500::X500Principal::~X500Principal ()
[inline, virtual]

6.741.2 Member Function Documentation

6.741.2.1 virtual void decaf::security::auth::x500::X500Principal::getEncoded (std::vector< unsigned char > & *output*) const [pure virtual]

6.741.2.2 virtual std::string decaf::security::auth::x500::X500Principal::getName () const [pure virtual]

Provides the name of this principal.

Returns

the name of this principal.

Implements **decaf::security::Principal** (p. 2412).

6.741.2.3 virtual int decaf::security::auth::x500::X500Principal::hashCode () const [pure virtual]

The documentation for this class was generated from the following file:

- src/main/decaf/security/auth/x500/X500Principal.h

6.742 decaf::security::cert::X509Certificate Class Reference

Base interface for all identity certificates.

```
#include <src/main/decaf/security/cert/X509Certificate.h>
```

Inheritance diagram for decaf::security::cert::X509Certificate:

Public Member Functions

- virtual ~X509Certificate ()
- virtual void **checkValidity** () const =0 throw (CertificateExpiredException, CertificateNotYetValidException)
- virtual void **checkValidity** (const decaf::util::Date &date) const =0 throw (CertificateExpiredException, CertificateNotYetValidException)
- virtual int **getBasicConstraints** () const =0
- virtual void **getIssuerUniqueID** (std::vector< bool > &output) const =0
- virtual const X500Principal * **getIssuerX500Principal** () const =0
- virtual void **getKeyUsage** (std::vector< unsigned char > &output) const =0
- virtual Date **getNotAfter** () const =0
- virtual Date **getNotBefore** () const =0

- virtual std::string **getSigAlgName** () const =0
- virtual std::string **getSigAlgOID** () const =0
- virtual void **getSigAlgParams** (std::vector< unsigned char > &output) const =0
- virtual void **getSignature** (std::vector< unsigned char > &output) const =0
- virtual void **getSubjectUniqueID** (std::vector< bool > &output) const =0
- virtual const X500Principal * **getSubjectX500Principal** () const =0
- virtual void **getTBSCertificate** (std::vector< unsigned char > &output) const =0 throw (CertificateEncodingException)
- virtual int **getVersion** () const =0

6.742.1 Detailed Description

Base interface for all identity certificates.

6.742.2 Constructor & Destructor Documentation

- 6.742.2.1** virtual decaf::security::cert::X509Certificate::~X509Certificate ()
[inline, virtual]

6.742.3 Member Function Documentation

- 6.742.3.1** virtual void decaf::security::cert::X509Certificate::checkValidity () const
throw (CertificateExpiredException, CertificateNotYetValidException)
[pure virtual]

Implemented in decaf::security_provider::unix::openssl::OpenSSLX509Certificate (p. 2291).

- 6.742.3.2** virtual void decaf::security::cert::X509Certificate::checkValidity (const decaf::util::Date & *date*) const throw (CertificateExpiredException, CertificateNotYetValidException) [pure virtual]

Implemented in decaf::security_provider::unix::openssl::OpenSSLX509Certificate (p. 2292).

- 6.742.3.3** virtual int decaf::security::cert::X509Certificate::getBasicConstraints () const [pure virtual]

Implemented in decaf::security_provider::unix::openssl::OpenSSLX509Certificate (p. 2292).

- 6.742.3.4** virtual void decaf::security::cert::X509Certificate::getIssuerUniqueID (std::vector< bool > & *output*) const [pure virtual]

Implemented in decaf::security_provider::unix::openssl::OpenSSLX509Certificate (p. 2292).

6.742.3.5 `virtual const X500Principal* decaf::security::cert::X509Certificate::getIssuerX500Principal () const [pure virtual]`

Implemented in `decaf::security_provider::unix::openssl::OpenSSLX509Certificate` (p. 2293).

6.742.3.6 `virtual void decaf::security::cert::X509Certificate::getKeyUsage (std::vector< unsigned char > & output) const [pure virtual]`

Implemented in `decaf::security_provider::unix::openssl::OpenSSLX509Certificate` (p. 2293).

6.742.3.7 `virtual Date decaf::security::cert::X509Certificate::getNotAfter () const [pure virtual]`

Implemented in `decaf::security_provider::unix::openssl::OpenSSLX509Certificate` (p. 2293).

6.742.3.8 `virtual Date decaf::security::cert::X509Certificate::getNotBefore () const [pure virtual]`

Implemented in `decaf::security_provider::unix::openssl::OpenSSLX509Certificate` (p. 2293).

6.742.3.9 `virtual std::string decaf::security::cert::X509Certificate::getSigAlgName () const [pure virtual]`

Implemented in `decaf::security_provider::unix::openssl::OpenSSLX509Certificate` (p. 2294).

6.742.3.10 `virtual std::string decaf::security::cert::X509Certificate::getSigAlgOID () const [pure virtual]`

Implemented in `decaf::security_provider::unix::openssl::OpenSSLX509Certificate` (p. 2294).

6.742.3.11 `virtual void decaf::security::cert::X509Certificate::getSigAlgParams (std::vector< unsigned char > & output) const [pure virtual]`

Implemented in `decaf::security_provider::unix::openssl::OpenSSLX509Certificate` (p. 2294).

6.742.3.12 `virtual void decaf::security::cert::X509Certificate::getSignature (std::vector< unsigned char > & output) const [pure virtual]`

Implemented in `decaf::security_provider::unix::openssl::OpenSSLX509Certificate` (p. 2294).

6.742.3.13 `virtual void decaf::security::cert::X509Certificate::getSubjectUniqueID (std::vector< bool > & output) const` [pure virtual]

Implemented in `decaf::security_provider::unix::openssl::OpenSSLX509Certificate` (p. 2294).

6.742.3.14 `virtual const X500Principal* decaf::security::cert::X509Certificate::getSubjectX500Principal () const` [pure virtual]

Implemented in `decaf::security_provider::unix::openssl::OpenSSLX509Certificate` (p. 2294).

6.742.3.15 `virtual void decaf::security::cert::X509Certificate::getTBSCertificate (std::vector< unsigned char > & output) const throw (CertificateEncodingException)` [pure virtual]

Implemented in `decaf::security_provider::unix::openssl::OpenSSLX509Certificate` (p. 2294).

6.742.3.16 `virtual int decaf::security::cert::X509Certificate::getVersion () const` [pure virtual]

Implemented in `decaf::security_provider::unix::openssl::OpenSSLX509Certificate` (p. 2295).

The documentation for this class was generated from the following file:

- `src/main/decaf/security/cert/X509Certificate.h`

6.743 activemq::commands::XATransactionId Class Reference

```
#include <src/main/activemq/commands/XATransactionId.h>
```

Inheritance diagram for `activemq::commands::XATransactionId`:

Public Types

- `typedef decaf::lang::PointerComparator< XATransactionId > COMPARATOR`

Public Member Functions

- `XATransactionId ()`
- `XATransactionId (const XATransactionId &other)`
- `virtual ~XATransactionId ()`
- `virtual unsigned char getDataStructureType () const`

Get the unique identifier that this object and its own Marshaler share.

- virtual **XATransactionId** * **cloneDataStructure** () const

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)

Copy the contents of the passed object into this object's members, overwriting any existing data.

- virtual std::string **toString** () const

*Returns a string containing the information for this **DataStructure** (p. 1372) such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** *value) const

*Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent.*

- virtual int **getFormatId** () const

- virtual void **setFormatId** (int **formatId**)

- virtual const std::vector< unsigned char > & **getGlobalTransactionId** () const

- virtual std::vector< unsigned char > & **getGlobalTransactionId** ()

- virtual void **setGlobalTransactionId** (const std::vector< unsigned char > &**globalTransactionId**)

- virtual const std::vector< unsigned char > & **getBranchQualifier** () const

- virtual std::vector< unsigned char > & **getBranchQualifier** ()

- virtual void **setBranchQualifier** (const std::vector< unsigned char > &**branchQualifier**)

- virtual int **compareTo** (const **XATransactionId** &value) const

- virtual bool **equals** (const **XATransactionId** &value) const

- virtual bool **operator==** (const **XATransactionId** &value) const

- virtual bool **operator<** (const **XATransactionId** &value) const

- **XATransactionId** & **operator=** (const **XATransactionId** &other)

Static Public Attributes

- static const unsigned char **ID_XATRANSACTIONID** = 112

Protected Attributes

- int **formatId**
- std::vector< unsigned char > **globalTransactionId**
- std::vector< unsigned char > **branchQualifier**

6.743.1 Member Typedef Documentation

6.743.1.1 `typedef decaf::lang::PointerComparator<XATransactionId>
activemq::commands::XATransactionId::COMPARATOR`

6.743.2 Constructor & Destructor Documentation

6.743.2.1 `activemq::commands::XATransactionId::XATransactionId ()`

6.743.2.2 `activemq::commands::XATransactionId::XATransactionId (const
XATransactionId & other)`

6.743.2.3 `virtual activemq::commands::XATransactionId::~~XATransactionId ()
[virtual]`

6.743.3 Member Function Documentation

6.743.3.1 `virtual XATransactionId* ac-
tivemq::commands::XATransactionId::cloneDataStructure
() const [virtual]`

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns

new copy of this object.

6.743.3.2 `virtual int activemq::commands::XATransactionId::compareTo (const
XATransactionId & value) const [virtual]`

6.743.3.3 `virtual void activemq::commands::XATransactionId::copyDataStructure (const
DataStructure * src) [virtual]`

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters

src - Source Object

6.743.3.4 `virtual bool activemq::commands::XATransactionId::equals (const
DataStructure * value) const [virtual]`

Compares the **DataStructure** (p. 1372) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns

true if DataStructure's are Equal.

- 6.743.3.5** virtual bool activemq::commands::XATransactionId::equals (const XATransactionId & *value*) const [virtual]
- 6.743.3.6** virtual const std::vector<unsigned char>& activemq::commands::XATransactionId::getBranchQualifier () const [virtual]
- 6.743.3.7** virtual std::vector<unsigned char>& activemq::commands::XATransactionId::getBranchQualifier () [virtual]
- 6.743.3.8** virtual unsigned char activemq::commands::XATransactionId::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns

new **DataSet** (p. 1372) type copy.

- 6.743.3.9 virtual int activemq::commands::XATransactionId::getFormatId ()
 const [virtual]

- 6.743.3.10 virtual const std::vector<unsigned char>& ac-
 tivemq::commands::XATransactionId::getGlobalTransactionId ()
 const [virtual]

- 6.743.3.11 virtual std::vector<unsigned char>& ac-
 tivemq::commands::XATransactionId::getGlobalTransactionId ()
 [virtual]

- 6.743.3.12 virtual bool activemq::commands::XATransactionId::operator< (const
 XATransactionId & *value*) const [virtual]

- 6.743.3.13 XATransactionId& activemq::commands::XATransactionId::operator= (const
 XATransactionId & *other*)

- 6.743.3.14 virtual bool activemq::commands::XATransactionId::operator== (const
 XATransactionId & *value*) const [virtual]

- 6.743.3.15 virtual void activemq::commands::XATransactionId::setBranchQualifier
 (const std::vector< unsigned char > & *branchQualifier*) [virtual]

- 6.743.3.16 virtual void activemq::commands::XATransactionId::setFormatId (int
 formatId) [virtual]

- 6.743.3.17 virtual void ac-
 tivemq::commands::XATransactionId::setGlobalTransactionId (const
 std::vector< unsigned char > & *globalTransactionId*) [virtual]

- 6.743.3.18 virtual std::string activemq::commands::XATransactionId::toString ()
 const [virtual]

Returns a string containing the information for this **DataStructure** (p.1372) such as its type and value of its elements.

Returns

formatted string useful for debugging.

6.743.4 Field Documentation

- 6.743.4.1 `std::vector<unsigned char> activemq::commands::XATransactionId::branchQualifier` [protected]
- 6.743.4.2 `int activemq::commands::XATransactionId::formatId` [protected]
- 6.743.4.3 `std::vector<unsigned char> activemq::commands::XATransactionId::globalTransactionId` [protected]
- 6.743.4.4 `const unsigned char activemq::commands::XATransactionId::ID_ - XATRANSACTIONID = 112` [static]

The documentation for this class was generated from the following file:

- `src/main/activemq/commands/XATransactionId.h`

6.744 activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller Class Reference

Marshaling code for Open Wire Format for `XATransactionIdMarshaller` (p. 3197).

```
#include <src/main/activemq/wireformat/openwire/marshal/v2/XATransactionIdMarshaller.h>
```

Inheritance diagram for `activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller`:

Public Member Functions

- `XATransactionIdMarshaller ()`
- `virtual ~XATransactionIdMarshaller ()`
- `virtual commands::DataStructure * createObject () const`
Creates a new instance of this marshalable type.
- `virtual unsigned char getDataStructureType () const`
Get the Data Structure Type that identifies this Marshaler.
- `virtual void tightUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Un-marshal an object instance from the data input stream.
- `virtual int tightMarshal1 (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, utils::BooleanStream *bs) throw (decaf::io::IOException)`
Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.744.1 Detailed Description

Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3197). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.744.2 Constructor & Destructor Documentation

6.744.2.1 **activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller::XATransactionIdMarshaller** () [inline]

6.744.2.2 virtual **activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller::~~XATransactionIdMarshaller** () [inline, virtual]

6.744.3 Member Function Documentation

6.744.3.1 virtual **commands::DataStructure*** **activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller::createObject** () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1331).

6.744.3.2 virtual unsigned char **activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller::getDataStructureType** () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements `activemq::wireformat::openwire::marshal::DataStreamMarshaller` (p. 1336).

6.744.3.3 `virtual void activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller::looseMarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataOutputStream * dataOut) throw (decaf::io::IOException)` [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller` (p. 3020).

6.744.3.4 `virtual void activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller::looseUnmarshal (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, decaf::io::DataInputStream * dataIn) throw (decaf::io::IOException)` [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller` (p. 3020).

6.744.3.5 `virtual int activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller::tightMarshal1 (OpenWireFormat * wireFormat, commands::DataStructure * dataStructure, utils::BooleanStream * bs) throw (decaf::io::IOException)` [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller` (p. 3021).

```
6.744.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker
dataStructure - Object to be marshaled
dataOut - BinaryReader that provides that data sink
bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller` (p. 3021).

```
6.744.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.
dataStructure - Object to be un-marshaled.
dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller** (p. 3022).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/XATransactionIdMarshaller.h

6.745 activemq::wireformat::openwire::marshal::v3::XATransactionIdMa Class Reference

Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3201).

```
#include <src/main/activemq/wireformat/openwire/marshal/v3/XATransactionIdMarshaller.h>
```

Inheritance diagram for activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller:

Public Member Functions

- **XATransactionIdMarshaller** ()
- virtual **~XATransactionIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (OpenWireFormat *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)
Write a object instance to data output stream.

- virtual void looseUnmarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataInputStream *dataIn) throw (decaf::io::IOException)

Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.745.1 Detailed Description

Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3201). NOTE! This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.745.2 Constructor & Destructor Documentation

6.745.2.1 activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller::XATransactionIdMarshaller () [inline]

6.745.2.2 virtual activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller::~~XATransactionIdMarshaller () [inline, virtual]

6.745.3 Member Function Documentation

6.745.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1331).

6.745.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1336).

6.745.3.3 virtual void activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller` (p. 3034).

6.745.3.4 virtual void activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller` (p. 3035).

6.745.3.5 virtual int activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller` (p. 3035).

```
6.745.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller` (p. 3036).

```
6.745.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller** (p. 3036).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v3/XATransactionIdMarshaller.h`

6.746 activemq::wireformat::openwire::marshal::v4::XATransactionIdMa Class Reference

Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3205).

```
#include <src/main/activemq/wireformat/openwire/marshal/v4/XATransactionIdMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller**:

Public Member Functions

- **XATransactionIdMarshaller** ()
- virtual **~XATransactionIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshal an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.746.1 Detailed Description

Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p.3205). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.746.2 Constructor & Destructor Documentation

6.746.2.1 activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller::XATransactionIdMarshaller () [inline]

6.746.2.2 virtual activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller::~XATransactionIdMarshaller () [inline, virtual]

6.746.3 Member Function Documentation

6.746.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1331).

6.746.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p.1336).

6.746.3.3 virtual void activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller` (p. 3027).

6.746.3.4 virtual void activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller` (p. 3027).

6.746.3.5 virtual int activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller` (p. 3028).

```
6.746.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller` (p. 3028).

```
6.746.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller** (p. 3029).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v4/XATransactionIdMarshaller.h`

6.747 activemq::wireformat::openwire::marshal::v1::XATransactionIdMa Class Reference

Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3209).

```
#include <src/main/activemq/wireformat/openwire/marshal/v1/XATransactionIdMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller**:

Public Member Functions

- **XATransactionIdMarshaller** ()
- virtual **~XATransactionIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshall an object instance from the data input stream.

- `virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)`

Write a object instance to data output stream.

6.747.1 Detailed Description

Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3209). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.747.2 Constructor & Destructor Documentation

6.747.2.1 `activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller::XATransactionIdMarshaller () [inline]`

6.747.2.2 `virtual activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller::~XATransactionIdMarshaller () [inline, virtual]`

6.747.3 Member Function Documentation

6.747.3.1 `virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller::createObject () const [virtual]`

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1331).

6.747.3.2 `virtual unsigned char activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller::getDataStructureType () const [virtual]`

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1336).

6.747.3.3 virtual void activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller` (p. 3023).

6.747.3.4 virtual void activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller` (p. 3024).

6.747.3.5 virtual int activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller` (p. 3024).

```
6.747.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller` (p. 3025).

```
6.747.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller** (p. 3025).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v1/XATransactionIdMarshaller.h`

6.748 activemq::wireformat::openwire::marshal::v5::XATransactionIdMa Class Reference

Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3213).

```
#include <src/main/activemq/wireformat/openwire/marshal/v5/XATransactionIdMarshaller.h>
```

Inheritance diagram for **activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller**:

Public Member Functions

- **XATransactionIdMarshaller** ()
- virtual **~XATransactionIdMarshaller** ()
- virtual **commands::DataStructure * createObject** () const
Creates a new instance of this marshalable type.
- virtual unsigned char **getDataStructureType** () const
Get the Data Structure Type that identifies this Marshaler.
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Un-marshal an object instance from the data input stream.
- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write the booleans that this object uses to a BooleanStream.
- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (**decaf::io::IOException**)
Write a object instance to data output stream.
- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (**decaf::io::IOException**)

Un-marshall an object instance from the data input stream.

- virtual void looseMarshal (OpenWireFormat *wireFormat, commands::DataStructure *dataStructure, decaf::io::DataOutputStream *dataOut) throw (decaf::io::IOException)

Write a object instance to data output stream.

6.748.1 Detailed Description

Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3213). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.748.2 Constructor & Destructor Documentation

6.748.2.1 activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller::XATransactionIdMarshaller () [inline]

6.748.2.2 virtual activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller::~XATransactionIdMarshaller () [inline, virtual]

6.748.3 Member Function Documentation

6.748.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1331).

6.748.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1336).

6.748.3.3 virtual void activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller` (p. 3031).

6.748.3.4 virtual void activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller` (p. 3031).

6.748.3.5 virtual int activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns

int value indicating the size of the marshaled object.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller` (p. 3032).

```
6.748.3.6 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller::tightMarshal2
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataOutputStream * dataOut,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Write a object instance to data output stream.

Parameters

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions

IOException if an error occurs during the marshal.

Reimplemented from `activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller` (p. 3032).

```
6.748.3.7 virtual void ac-
tivemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller::tightUnmarshal
( OpenWireFormat * wireFormat, commands::DataStructure
* dataStructure, decaf::io::DataInputStream * dataIn,
utils::BooleanStream * bs ) throw ( decaf::io::IOException ) [virtual]
```

Un-marshall an object instance from the data input stream.

Parameters

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller** (p. 3033).

The documentation for this class was generated from the following file:

- `src/main/activemq/wireformat/openwire/marshal/v5/XATransactionIdMarshaller.h`

Chapter 7

File Documentation

7.1 src/main/activemq/cmsutil/CachedConsumer.h File Reference

```
#include <cms/MessageConsumer.h>
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::cmsutil::CachedConsumer**
A cached message consumer contained within a pooled session.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::cmsutil**

7.2 src/main/activemq/cmsutil/CachedProducer.h File Reference

```
#include <cms/MessageProducer.h>
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::cmsutil::CachedProducer**
A cached message producer contained within a pooled session.

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::cmsutil**

7.3 src/main/activemq/cmsutil/CmsAccessor.h File Reference

```
#include <cms/ConnectionFactory.h>
#include <activemq/cmsutil/ResourceLifecycleManager.h>
#include <activemq/util/Config.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
```

Data Structures

- class **activemq::cmsutil::CmsAccessor**

*Base class for **activemq.cmsutil.CmsTemplate** (p. 969) and other CMS-accessing gateway helpers, defining common properties such as the CMS **cms.ConnectionFactory** (p. 1103) to operate on.*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::cmsutil**

7.4 src/main/activemq/cmsutil/CmsDestinationAccessor.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/cmsutil/CmsAccessor.h>
#include <activemq/cmsutil/DynamicDestinationResolver.h>
```

Data Structures

- class **activemq::cmsutil::CmsDestinationAccessor**

*Extends the **CmsAccessor** (p. 954) to add support for resolving destination names.*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::cmsutil**

7.5 src/main/activemq/cmsutil/CmsTemplate.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/cmsutil/CmsDestinationAccessor.h>
#include <activemq/cmsutil/SessionCallback.h>
#include <activemq/cmsutil/ProducerCallback.h>
#include <activemq/cmsutil/SessionPool.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
#include <cms/ConnectionFactory.h>
#include <cms/DeliveryMode.h>
#include <string>
```

Data Structures

- class **activemq::cmsutil::CmsTemplate**
CmsTemplate (p. 969) simplifies performing synchronous CMS operations.
- class **activemq::cmsutil::CmsTemplate::ProducerExecutor**
- class **activemq::cmsutil::CmsTemplate::ResolveProducerExecutor**
- class **activemq::cmsutil::CmsTemplate::SendExecutor**
- class **activemq::cmsutil::CmsTemplate::ReceiveExecutor**
- class **activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::cmsutil**

7.6 src/main/activemq/cmsutil/DestinationResolver.h File Reference

```
#include <cms/Session.h>
```

```
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::cmsutil::DestinationResolver**
*Resolves a CMS destination name to a *Destination*.*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::cmsutil**

7.7 src/main/activemq/cmsutil/DynamicDestinationResolver.h File Reference

```
#include <activemq/cmsutil/DestinationResolver.h>  
#include <cms/Session.h>  
#include <decaf/util/StlMap.h>  
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::cmsutil::DynamicDestinationResolver**
*Resolves a CMS destination name to a *Destination*.*
- class **activemq::cmsutil::DynamicDestinationResolver::SessionResolver**
Manages maps of names to topics and queues for a single session.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::cmsutil**

7.8 src/main/activemq/cmsutil/MessageCreator.h File Reference

```
#include <cms/Session.h>
```

```
#include <cms/Message.h>
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::cmsutil::MessageCreator**
Creates the user-defined message to be sent by the `CmsTemplate` (p. 969).

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::cmsutil**

7.9 src/main/activemq/cmsutil/PooledSession.h File Reference

```
#include <cms/Session.h>
#include <decaf/util/StlMap.h>
#include <activemq/cmsutil/CachedProducer.h>
#include <activemq/cmsutil/CachedConsumer.h>
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::cmsutil::PooledSession**
A pooled session object that wraps around a delegate session.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::cmsutil**

7.10 src/main/activemq/cmsutil/ProducerCallback.h File Reference

```
#include <cms/Session.h>
```

```
#include <cms/MessageProducer.h>
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::cmsutil::ProducerCallback**
Callback for sending a message to a CMS destination.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::cmsutil**

7.11 src/main/activemq/cmsutil/ResourceLifecycleManager.h File Reference

```
#include <cms/Connection.h>
#include <cms/Session.h>
#include <cms/Destination.h>
#include <cms/MessageProducer.h>
#include <cms/MessageConsumer.h>
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::cmsutil::ResourceLifecycleManager**
Manages the lifecycle of a set of CMS resources.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::cmsutil**

7.12 src/main/activemq/cmsutil/SessionCallback.h File Reference

```
#include <cms/Session.h>
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::cmsutil::SessionCallback**
Callback for executing any number of operations on a provided CMS Session.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::cmsutil**

7.13 src/main/activemq/cmsutil/SessionPool.h File Reference

```
#include <activemq/cmsutil/PooledSession.h>
#include <decaf/util/concurrent/Mutex.h>
#include <cms/Connection.h>
#include <list>
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::cmsutil::SessionPool**
A pool of CMS sessions from the same connection and with the same acknowledge mode.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::cmsutil**

7.14 src/main/activemq/commands/ActiveMQBlobMessage.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ActiveMQMessageTemplate.h>
#include <cms/Message.h>
#include <string>
#include <memory>
```

Data Structures

- class `activemq::commands::ActiveMQBlobMessage`

Namespaces

- namespace `activemq`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `activemq::commands`

7.15 src/main/activemq/commands/ActiveMQBytesMessage.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ActiveMQMessageTemplate.h>
#include <decaf/io/ByteArrayOutputStream.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <cms/BytesMessage.h>
#include <vector>
#include <string>
#include <memory>
```

Data Structures

- class `activemq::commands::ActiveMQBytesMessage`

Namespaces

- namespace `activemq`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.16 src/main/activemq/commands/ActiveMQDestination.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/util/ActiveMQProperties.h>
#include <cms/Destination.h>
#include <decaf/lang/Pointer.h>
#include <vector>
#include <string>
#include <map>
```

Data Structures

- class **activemq::commands::ActiveMQDestination**
- struct **activemq::commands::ActiveMQDestination::DestinationFilter**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.17 src/main/activemq/commands/ActiveMQMapMessage.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ActiveMQMessageTemplate.h>
#include <activemq/util/PrimitiveMap.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <cms/MapMessage.h>
#include <vector>
#include <string>
#include <memory>
```

Data Structures

- class `activemq::commands::ActiveMQMapMessage`

Namespaces

- namespace `activemq`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `activemq::commands`

7.18 `src/main/activemq/commands/ActiveMQMessage.h` File Reference

```
#include <cms/Message.h>
#include <activemq/util/Config.h>
#include <activemq/commands/ActiveMQMessageTemplate.h>
```

Data Structures

- class `activemq::commands::ActiveMQMessage`

Namespaces

- namespace `activemq`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `activemq::commands`

7.19 `src/main/activemq/commands/ActiveMQMessageTemplate.h` File Reference

```
#include <cms/DeliveryMode.h>
#include <activemq/util/Config.h>
#include <activemq/commands/Message.h>
#include <activemq/core/ActiveMQAckHandler.h>
#include <activemq/wireformat/openwire/utils/MessagePropertyInterceptor.h>
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <activemq/util/CMSExceptionSupport.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <cms/IllegalStateException.h>
```


7.20 src/main/activemq/commands/ActiveMQObjectMessage.h File Reference 3233

```
#include <cms/MessageFormatException.h>
#include <cms/MessageNotReadableException.h>
#include <cms/MessageNotWriteableException.h>
```

Data Structures

- class **activemq::commands::ActiveMQMessageTemplate**< T >

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.20 src/main/activemq/commands/ActiveMQObjectMessage.h File Reference

```
#include <activemq/commands/ActiveMQMessageTemplate.h>
#include <cms/ObjectMessage.h>
#include <activemq/util/Config.h>
#include <memory>
```

Data Structures

- class **activemq::commands::ActiveMQObjectMessage**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.21 src/main/activemq/commands/ActiveMQQueue.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <decaf/lang/Exception.h>
```

```
#include <cms/Queue.h>
#include <vector>
#include <string>
#include <memory>
```

Data Structures

- class `activemq::commands::ActiveMQQueue`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::commands`

7.22 src/main/activemq/commands/ActiveMQStreamMessage.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ActiveMQMessage.h>
#include <activemq/commands/ActiveMQMessageTemplate.h>
#include <cms/StreamMessage.h>
#include <cms/MessageNotWriteableException.h>
#include <cms/MessageNotReadableException.h>
#include <cms/MessageFormatException.h>
#include <cms/MessageEOFException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/ByteArrayOutputStream.h>
#include <string>
#include <memory>
```

Data Structures

- class `activemq::commands::ActiveMQStreamMessage`

Namespaces

- namespace `activemq`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.23 src/main/activemq/commands/ActiveMQTempDestination.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <cms/Closeable.h>
#include <vector>
#include <string>
```

Data Structures

- class **activemq::commands::ActiveMQTempDestination**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::core**
- namespace **activemq::commands**

7.24 src/main/activemq/commands/ActiveMQTempQueue.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ActiveMQTempDestination.h>
#include <cms/TemporaryQueue.h>
#include <vector>
#include <string>
#include <memory>
```

Data Structures

- class **activemq::commands::ActiveMQTempQueue**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.25 src/main/activemq/commands/ActiveMQTempTopic.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ActiveMQTempDestination.h>
#include <cms/TemporaryTopic.h>
#include <vector>
#include <string>
#include <memory>
```

Data Structures

- class **activemq::commands::ActiveMQTempTopic**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.26 src/main/activemq/commands/ActiveMQTextMessage.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ActiveMQMessageTemplate.h>
#include <cms/TextMessage.h>
#include <vector>
#include <string>
#include <memory>
```

Data Structures

- class **activemq::commands::ActiveMQTextMessage**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.27 src/main/activemq/commands/ActiveMQTopic.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <decaf/lang/Exception.h>
#include <cms/Topic.h>
#include <vector>
#include <string>
#include <memory>
```

Data Structures

- class **activemq::commands::ActiveMQTopic**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.28 src/main/activemq/commands/BaseCommand.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/Command.h>
```

Data Structures

- class **activemq::commands::BaseCommand**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.29 src/main/activemq/commands/BaseDataStructure.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <string>
#include <sstream>
```

Data Structures

- class **activemq::commands::BaseDataStructure**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::commands**

7.30 src/main/activemq/commands/BooleanExpression.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::commands::BooleanExpression**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.31 src/main/activemq/commands/BrokerError.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/BaseCommand.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::BrokerError**
This class represents an Exception sent from the Broker.
- struct **activemq::commands::BrokerError::StackTraceElement**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.32 src/main/activemq/commands/BrokerId.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::BrokerId**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.33 src/main/activemq/commands/BrokerInfo.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/BrokerId.h>
#include <activemq/commands/BrokerInfo.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::BrokerInfo**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.34 src/main/activemq/commands/Command.h File Reference

```
#include <string>
#include <activemq/util/Config.h>
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::commands::Command**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::state**
- namespace **activemq::commands**

7.35 src/main/activemq/commands/ConnectionControl.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::ConnectionControl**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.36 src/main/activemq/commands/ConnectionError.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/BrokerError.h>
#include <activemq/commands/ConnectionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class `activemq::commands::ConnectionError`

Namespaces

- namespace `activemq`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `activemq::commands`

7.37 `src/main/activemq/commands/ConnectionId.h` File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class `activemq::commands::ConnectionId`

Namespaces

- namespace `activemq`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `activemq::commands`

7.38 `src/main/activemq/commands/ConnectionInfo.h` File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/BrokerId.h>
#include <activemq/commands/ConnectionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
```

```
#include <vector>
```

Data Structures

- class `activemq::commands::ConnectionInfo`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::commands`

7.39 src/main/activemq/commands/ConsumerControl.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class `activemq::commands::ConsumerControl`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::commands`

7.40 src/main/activemq/commands/ConsumerId.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/commands/SessionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Comparable.h>
```

```
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::ConsumerId**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.41 src/main/activemq/commands/ConsumerInfo.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/BooleanExpression.h>
#include <activemq/commands/BrokerId.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::ConsumerInfo**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.42 src/main/activemq/commands/ControlCommand.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class `activemq::commands::ControlCommand`

Namespaces

- namespace `activemq`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `activemq::commands`

7.43 src/main/activemq/commands/DataArrayResponse.h File Reference

```
#include <activemq/commands/DataStructure.h>
#include <activemq/commands/Response.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class `activemq::commands::DataArrayResponse`

Namespaces

- namespace `activemq`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `activemq::commands`

7.44 src/main/activemq/commands/DataResponse.h File Reference

```
#include <activemq/commands/DataSet.h>
#include <activemq/commands/Response.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class `activemq::commands::DataResponse`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::commands`

7.45 src/main/activemq/commands/DataStructure.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/wireformat/MarshalAware.h>
```

Data Structures

- class `activemq::commands::DataSet`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::commands`

7.46 src/main/activemq/commands/DestinationInfo.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
```

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/BrokerId.h>
#include <activemq/commands/ConnectionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class `activemq::commands::DestinationInfo`

Namespaces

- namespace `activemq`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `activemq::commands`

7.47 src/main/activemq/commands/DiscoveryEvent.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class `activemq::commands::DiscoveryEvent`

Namespaces

- namespace `activemq`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `activemq::commands`

7.48 src/main/activemq/commands/ExceptionResponse.h File Reference

```
#include <activemq/commands/BrokerError.h>
#include <activemq/commands/Response.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class `activemq::commands::ExceptionResponse`

Namespaces

- namespace `activemq`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `activemq::commands`

7.49 src/main/activemq/commands/FlushCommand.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class `activemq::commands::FlushCommand`

Namespaces

- namespace `activemq`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `activemq::commands`

7.50 src/main/activemq/commands/IntegerResponse.h File Reference

```
#include <activemq/commands/Response.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class `activemq::commands::IntegerResponse`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::commands`

7.51 src/main/activemq/commands/JournalQueueAck.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/commands/MessageAck.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class `activemq::commands::JournalQueueAck`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::commands`

7.52 src/main/activemq/commands/JournalTopicAck.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/commands/MessageId.h>
#include <activemq/commands/TransactionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class `activemq::commands::JournalTopicAck`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::commands`

7.53 src/main/activemq/commands/JournalTrace.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class `activemq::commands::JournalTrace`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.54 src/main/activemq/commands/JournalTransaction.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/commands/TransactionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::JournalTransaction**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.55 src/main/activemq/commands/KeepAliveInfo.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::KeepAliveInfo**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.56 src/main/activemq/commands/LastPartialCommand.h File Reference

```
#include <activemq/commands/PartialCommand.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::LastPartialCommand**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.57 src/main/activemq/commands/LocalTransactionId.h File Reference

```
#include <activemq/commands/ConnectionId.h>
#include <activemq/commands/TransactionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::LocalTransactionId**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.58 src/main/activemq/commands/Message.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/BrokerId.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/DataSet.h>
#include <activemq/commands/DataSet.h>
#include <activemq/commands/MessageId.h>
#include <activemq/commands/ProducerId.h>
#include <activemq/commands/TransactionId.h>
#include <activemq/core/ActiveMQAckHandler.h>
#include <activemq/util/Config.h>
#include <activemq/util/PrimitiveMap.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::Message**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::core**
- namespace **activemq::commands**

7.59 src/main/cms/Message.h File Reference

```
#include <cms/Config.h>
#include <cms/Destination.h>
#include <cms/DeliveryMode.h>
```

```
#include <cms/CMSException.h>
#include <cms/IllegalStateException.h>
#include <cms/MessageFormatException.h>
#include <cms/MessageNotWriteableException.h>
```

Data Structures

- class **cms::Message**
Root of all messages.

Namespaces

- namespace **cms**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.60 src/main/activemq/commands/MessageAck.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/MessageId.h>
#include <activemq/commands/TransactionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::MessageAck**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.61 src/main/activemq/commands/MessageDispatch.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/Message.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class `activemq::commands::MessageDispatch`

Namespaces

- namespace `activemq`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `activemq::commands`

7.62 src/main/activemq/commands/MessageDispatchNotification.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/MessageId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class `activemq::commands::MessageDispatchNotification`

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.63 src/main/activemq/commands/MessageId.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/commands/ProducerId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::MessageId**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.64 src/main/activemq/commands/MessagePull.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/MessageId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```


Data Structures

- class `activemq::commands::MessagePull`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::commands`

7.65 src/main/activemq/commands/NetworkBridgeFilter.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/commands/BrokerId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class `activemq::commands::NetworkBridgeFilter`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::commands`

7.66 src/main/activemq/commands/PartialCommand.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class `activemq::commands::PartialCommand`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::commands`

7.67 `src/main/activemq/commands/ProducerAck.h` File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/ProducerId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class `activemq::commands::ProducerAck`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::commands`

7.68 `src/main/activemq/commands/ProducerId.h` File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/commands/SessionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/Pointer.h>
#include <string>
```

```
#include <vector>
```

Data Structures

- class `activemq::commands::ProducerId`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::commands`

7.69 src/main/activemq/commands/ProducerInfo.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/BrokerId.h>
#include <activemq/commands/ProducerId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class `activemq::commands::ProducerInfo`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::commands`

7.70 src/main/activemq/commands/RemoveInfo.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/DataStructure.h>
```

```
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::RemoveInfo**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.71 src/main/activemq/commands/RemoveSubscriptionInfo.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/ConnectionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::RemoveSubscriptionInfo**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.72 src/main/activemq/commands/ReplayCommand.h File Reference

```
#include <activemq/commands/BaseCommand.h>
```

```
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::ReplayCommand**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.73 src/main/activemq/commands/Response.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::Response**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**

7.74 src/main/activemq/commands/SessionId.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/commands/ConnectionId.h>
```

```
#include <activemq/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::SessionId**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.75 src/main/activemq/commands/SessionInfo.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/SessionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::SessionInfo**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.76 src/main/activemq/commands/ShutdownInfo.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class `activemq::commands::ShutdownInfo`

Namespaces

- namespace `activemq`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `activemq::commands`

7.77 src/main/activemq/commands/SubscriptionInfo.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class `activemq::commands::SubscriptionInfo`

Namespaces

- namespace `activemq`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `activemq::commands`

7.78 src/main/activemq/commands/TransactionId.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class `activemq::commands::TransactionId`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::commands`

7.79 src/main/activemq/commands/TransactionInfo.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/ConnectionId.h>
#include <activemq/commands/TransactionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class `activemq::commands::TransactionInfo`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.80 src/main/activemq/commands/WireFormatInfo.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/BaseCommand.h>
#include <activemq/util/PrimitiveMap.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <vector>
```

Data Structures

- class **activemq::commands::WireFormatInfo**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.81 src/main/activemq/commands/XATransactionId.h File Reference

```
#include <activemq/commands/TransactionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::XATransactionId**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.82 src/main/activemq/core/ActiveMQAckHandler.h File Reference

```
#include <cms/CMSException.h>
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::core::ActiveMQAckHandler**
Interface class that is used to give CMS Messages an interface to Ack themselves with.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::commands**
- namespace **activemq::core**

7.83 src/main/activemq/core/ActiveMQConnection.h File Reference

```
#include <cms/Connection.h>
#include <activemq/util/Config.h>
#include <activemq/core/ActiveMQConnectionSupport.h>
#include <activemq/core/ActiveMQConnectionMetaData.h>
#include <activemq/core/Dispatcher.h>
#include <activemq/commands/ActiveMQTempDestination.h>
#include <activemq/commands/BrokerInfo.h>
#include <activemq/commands/ConnectionInfo.h>
#include <activemq/commands/ConsumerInfo.h>
#include <activemq/commands/ProducerInfo.h>
#include <activemq/commands/LocalTransactionId.h>
#include <activemq/commands/WireFormatInfo.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/transport/TransportListener.h>
```

```
#include <decaf/util/Properties.h>
#include <decaf/util/StlMap.h>
#include <decaf/util/StlSet.h>
#include <decaf/util/concurrent/atomic/AtomicBoolean.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
#include <string>
#include <memory>
```

Data Structures

- class **activemq::core::ActiveMQConnection**
Concrete connection used for all connectors to the ActiveMQ broker.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::core**

7.84 src/main/activemq/core/ActiveMQConnectionFactory.h File Reference

```
#include <activemq/util/Config.h>
#include <cms/ConnectionFactory.h>
#include <cms/Connection.h>
```

Data Structures

- class **activemq::core::ActiveMQConnectionFactory**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::core**

7.85 src/main/activemq/core/ActiveMQConnectionMetaData.h File Reference

```
#include <activemq/util/Config.h>
#include <cms/ConnectionMetaData.h>
```

Data Structures

- class **activemq::core::ActiveMQConnectionMetaData**

*This class houses all the various settings and information that is used by an instance of an **ActiveMQConnection** (p. 202) class.*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::core**

7.86 src/main/activemq/core/ActiveMQConnectionSupport.h File Reference

```
#include <cms/ExceptionListener.h>
#include <activemq/util/Config.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/transport/Transport.h>
#include <activemq/transport/TransportListener.h>
#include <activemq/util/LongSequenceGenerator.h>
#include <decaf/util/Properties.h>
#include <decaf/lang/Exception.h>
#include <decaf/lang/Pointer.h>
#include <memory>
```

Data Structures

- class **activemq::core::ActiveMQConnectionSupport**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::core**

7.87 src/main/activemq/core/ActiveMQConstants.h File Reference

```
#include <string>
#include <map>
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::core::ActiveMQConstants**

Class holding constant values for various ActiveMQ specific things Each constant is defined as an enumeration and has functions that convert back an forth between string and enum values.

- class **activemq::core::ActiveMQConstants::StaticInitializer**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::core**

7.88 src/main/activemq/core/ActiveMQConsumer.h File Reference

```
#include <cms/MessageConsumer.h>
#include <cms/MessageListener.h>
#include <cms/Message.h>
#include <cms/CMSException.h>
#include <activemq/util/Config.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/commands/ConsumerInfo.h>
#include <activemq/commands/MessageAck.h>
#include <activemq/commands/MessageDispatch.h>
#include <activemq/core/ActiveMQTransactionContext.h>
#include <activemq/core/Dispatcher.h>
```

```
#include <activemq/core/MessageDispatchChannel.h>
#include <decaf/util/concurrent/atomic/AtomicBoolean.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/StlQueue.h>
#include <decaf/util/concurrent/Mutex.h>
#include <memory>
```

Data Structures

- class `activemq::core::ActiveMQConsumer`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::core`

7.89 `src/main/activemq/core/ActiveMQProducer.h` File Reference

```
#include <cms/MessageProducer.h>
#include <cms/Message.h>
#include <cms/Destination.h>
#include <cms/DeliveryMode.h>
#include <activemq/util/Config.h>
#include <activemq/util/MemoryUsage.h>
#include <activemq/commands/ProducerInfo.h>
#include <activemq/commands/ProducerAck.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <memory>
```

Data Structures

- class `activemq::core::ActiveMQProducer`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::core**

7.90 src/main/activemq/core/ActiveMQSession.h File Reference

```
#include <cms/Session.h>
#include <cms/ExceptionListener.h>
#include <activemq/util/Config.h>
#include <activemq/util/Usage.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/core/ActiveMQTransactionContext.h>
#include <activemq/commands/ActiveMQTempDestination.h>
#include <activemq/commands/SessionInfo.h>
#include <activemq/commands/ConsumerInfo.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/ProducerId.h>
#include <activemq/commands/TransactionId.h>
#include <activemq/core/Dispatcher.h>
#include <activemq/core/MessageDispatchChannel.h>
#include <activemq/util/LongSequenceGenerator.h>
#include <decaf/util/StlMap.h>
#include <decaf/util/StlQueue.h>
#include <decaf/util/Properties.h>
#include <string>
#include <memory>
```

Data Structures

- class **activemq::core::ActiveMQSession**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::core**

7.91 src/main/activemq/core/ActiveMQSessionExecutor.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/core/MessageDispatchChannel.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/MessageDispatch.h>
#include <activemq/threads/Task.h>
#include <activemq/threads/TaskRunner.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::core::ActiveMQSessionExecutor**

Delegate dispatcher for a single session.

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::core**

7.92 src/main/activemq/core/ActiveMQTransactionContext.h File Reference

```
#include <memory>
#include <cms/Message.h>
#include <activemq/util/Config.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/commands/LocalTransactionId.h>
#include <activemq/core/Synchronization.h>
#include <activemq/util/LongSequenceGenerator.h>
#include <decaf/lang/exceptions/InvalidStateException.h>
#include <decaf/util/StlSet.h>
#include <decaf/util/Properties.h>
#include <decaf/util/concurrent/Mutex.h>
```


Data Structures

- class **activemq::core::ActiveMQTransactionContext**

Transaction Management class, hold messages that are to be redelivered upon a request to roll-back.

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::core**

7.93 src/main/activemq/core/DispatchData.h File Reference

```
#include <stdlib.h>
#include <memory>
#include <activemq/util/Config.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/Message.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::core::DispatchData**

Simple POJO that contains the information necessary to route a message to a specified consumer.

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::core**

7.94 src/main/activemq/core/Dispatcher.h File Reference

```
#include <activemq/commands/MessageDispatch.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::core::Dispatcher**

Interface for an object responsible for dispatching messages to consumers.

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::core**

7.95 src/main/activemq/core/MessageDispatchChannel.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/MessageDispatch.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/StlQueue.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::core::MessageDispatchChannel**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::core**

7.96 src/main/activemq/core/Synchronization.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/exceptions/ActiveMQException.h>
```

Data Structures

- class `activemq::core::Synchronization`

*Transacted Object **Synchronization** (p. 2945), used to sync the events of a Transaction with the items in the Transaction.*

Namespaces

- namespace `activemq`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `activemq::core`

7.97 src/main/activemq/exceptions/ActiveMQException.h File Reference

```
#include <activemq/util/Config.h>
#include <cms/CMSException.h>
#include <decaf/lang/Exception.h>
#include <activemq/exceptions/ExceptionDefines.h>
#include <stdarg.h>
#include <sstream>
```

Data Structures

- class `activemq::exceptions::ActiveMQException`

Namespaces

- namespace `activemq`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `activemq::exceptions`

7.98 src/main/activemq/exceptions/BrokerException.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/commands/BrokerError.h>
#include <sstream>
```

Data Structures

- class `activemq::exceptions::BrokerException`

Namespaces

- namespace `activemq`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `activemq::exceptions`

7.99 src/main/activemq/exceptions/ExceptionDefines.h File Reference

Defines

- `#define AMQ_CATCH_RETHROW(type)`
Macro for catching and re-throwing an exception of a given type.
- `#define AMQ_CATCH_EXCEPTION_CONVERT(sourceType, targetType)`
Macro for catching an exception of one type and then re-throwing as another type.
- `#define AMQ_CATCHALL_THROW(type)`
A catch-all that throws a known exception.
- `#define AMQ_CATCHALL_NOTHROW()`
A catch-all that does not throw an exception, one use would be to catch any exception in a destructor and mark it, but not throw so that cleanup would continue as normal.
- `#define AMQ_CATCH_NOTHROW(type)`
Macro for catching and re-throwing an exception of a given type.

7.99.1 Define Documentation

7.99.1.1 `#define AMQ_CATCH_EXCEPTION_CONVERT(sourceType, targetType)`

Value:

```
catch( sourceType& ex ){ \
    targetType target( ex ); \
    target.setMark( __FILE__, __LINE__ ); \
    throw target; \
}
```

Macro for catching an exception of one type and then re-throwing as another type.

Parameters

sourceType the type of the exception to be caught.

targetType the type of the exception to be thrown.

Referenced by `activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshalObjectArray()`, `activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalObjectArray1()`, and `activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalObjectArray2()`.

7.99.1.2 #define AMQ_CATCH_NOTHROW(type)**Value:**

```
catch( type& ex ){ \
    ex.setMark( __FILE__, __LINE__ ); \
}
```

Macro for catching and re-throwing an exception of a given type.

Parameters

type The type of the exception to throw (e.g. `ActiveMQException`).

7.99.1.3 #define AMQ_CATCH_RETHROW(type)**Value:**

```
catch( type& ex ){ \
    ex.setMark( __FILE__, __LINE__ ); \
    throw ex; \
}
```

Macro for catching and re-throwing an exception of a given type.

Parameters

type The type of the exception to throw (e.g. `ActiveMQException`).

Referenced by `activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshalObjectArray()`, `activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalObjectArray1()`, and `activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalObjectArray2()`.

7.99.1.4 #define AMQ_CATCHALL_NOTHROW()**Value:**

```
catch( ... ){ \
    activemq::exceptions::ActiveMQException ex( __FILE__, __LINE__, \
        "caught unknown exception, not rethrowing" ); \
}
```

A catch-all that does not throw an exception, one use would be to catch any exception in a destructor and mark it, but not throw so that cleanup would continue as normal.

7.99.1.5 `#define AMQ_CATCHALL_THROW(type)`

Value:

```
catch( ... ){ \
    type ex( __FILE__, __LINE__, \
        "caught unknown exception" ); \
    throw ex; \
}
```

A catch-all that throws a known exception.

Parameters

type the type of exception to be thrown.

Referenced by `activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshalObjectArray()`, `activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalObjectArray1()`, and `activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalObjectArray2()`.

7.100 `src/main/decaf/lang/exceptions/ExceptionDefines.h` File Reference

Defines

- `#define DECAF_CATCH_RETHROW(type)`
Macro for catching and rethrowing an exception of a given type.
- `#define DECAF_CATCH_EXCEPTION_CONVERT(sourceType, targetType)`
Macro for catching an exception of one type and then rethrowing as another type.
- `#define DECAF_CATCHALL_THROW(type)`
A catch-all that throws a known exception.
- `#define DECAF_CATCHALL_NOTHROW()`
A catch-all that does not throw an exception, one use would be to catch any exception in a destructor and mark it, but not throw so that cleanup would continue as normal.
- `#define DECAF_CATCH_NOTHROW(type)`
Macro for catching and rethrowing an exception of a given type.

7.100.1 Define Documentation

7.100.1.1 `#define DECAF_CATCH_EXCEPTION_CONVERT(sourceType, targetType)`

Value:

```
catch( sourceType& ex ){ \
    targetType target( &ex ); \
    target.setMark( __FILE__, __LINE__ ); \
    throw target; \
}
```

Macro for catching an exception of one type and then rethrowing as another type.

Parameters

sourceType the type of the exception to be caught.

targetType the type of the exception to be thrown.

Referenced by decaf::util::PriorityQueue< E >::add().

7.100.1.2 #define DECAF_CATCH_NOTHROW(*type*)

Value:

```
catch( type& ex ){ \
    ex.setMark( __FILE__, __LINE__ ); \
}
```

Macro for catching and rethrowing an exception of a given type.

Parameters

type The type of the exception to throw (e.g. Exception).

Referenced by decaf::io::FilterInputStream::~~FilterInputStream(), decaf::io::FilterOutputStream::~~FilterOutputStream(), and decaf::util::logging::StreamHandler::~~StreamHandler().

7.100.1.3 #define DECAF_CATCH_RETHROW(*type*)

Value:

```
catch( type& ex ){ \
    ex.setMark( __FILE__, __LINE__ ); \
    throw ex; \
}
```

Macro for catching and rethrowing an exception of a given type.

Parameters

type The type of the exception to throw (e.g. Exception).

Referenced by decaf::util::PriorityQueue< E >::add(), decaf::io::FilterInputStream::available(), decaf::io::FilterOutputStream::close(), decaf::io::FilterInputStream::close(), decaf::io::FilterOutputStream::flush(), decaf::util::concurrent::Lock::lock(), decaf::util::concurrent::Lock::Lock(), decaf::util::logging::StreamHandler::publish(), decaf::io::FilterInputStream::read(), decaf::io::FilterInputStream::reset(), decaf::io::FilterInputStream::skip(), decaf::util::concurrent::Lock::unlock(), decaf::io::FilterOutputStream::write(), and decaf::util::concurrent::Lock::~~Lock().

7.100.1.4 #define DECAF_CATCHALL_NOTHROW()

Value:

```
catch( ... ){ \
    lang::Exception ex( __FILE__, __LINE__, \
        "caught unknown exception, not rethrowing" ); \
}
```

A catch-all that does not throw an exception, one use would be to catch any exception in a destructor and mark it, but not throw so that cleanup would continue as normal.

Referenced by `decaf::io::FilterInputStream::mark()`, `decaf::io::FilterInputStream::markSupported()`, `decaf::io::FilterInputStream::~~FilterInputStream()`, `decaf::io::FilterOutputStream::~~FilterOutputStream()`, and `decaf::util::logging::StreamHandler::~~StreamHandler()`.

7.100.1.5 `#define DECAF_CATCHALL_THROW(type)`

Value:

```
catch( ... ){ \
    type ex( __FILE__, __LINE__, \
        "caught unknown exception" ); \
    throw ex; \
}
```

A catch-all that throws a known exception.

Parameters

type the type of exception to be thrown.

Referenced by `decaf::util::PriorityQueue< E >::add()`, `decaf::io::FilterInputStream::available()`, `decaf::io::FilterOutputStream::close()`, `decaf::io::FilterInputStream::close()`, `decaf::io::FilterOutputStream::flush()`, `decaf::util::concurrent::Lock::lock()`, `decaf::util::concurrent::Lock::Lock()`, `decaf::util::logging::StreamHandler::publish()`, `decaf::io::FilterInputStream::read()`, `decaf::io::FilterInputStream::reset()`, `decaf::io::FilterInputStream::skip()`, `decaf::util::concurrent::Lock::unlock()`, `decaf::io::FilterOutputStream::write()`, and `decaf::util::concurrent::Lock::~~Lock()`.

7.101 `src/main/activemq/io/LoggingInputStream.h` File Reference

```
#include <activemq/util/Config.h>
#include <decaf/io/FilterInputStream.h>
#include <decaf/util/logging/LoggerDefines.h>
#include <decaf/lang/exceptions/NullPointerException.h>
```

Data Structures

- class `activemq::io::LoggingInputStream`

Namespaces

- namespace `activemq`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::io**

7.102 src/main/activemq/io/LoggingOutputStream.h File Reference

```
#include <activemq/util/Config.h>
#include <decaf/io/FilterOutputStream.h>
#include <decaf/util/logging/LoggerDefines.h>
```

Data Structures

- class **activemq::io::LoggingOutputStream**
OutputStream filter that just logs the data being written.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::io**

7.103 src/main/activemq/library/ActiveMQCPP.h File Reference

```
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::library::ActiveMQCPP**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::library**

7.104 src/main/activemq/state/CommandVisitor.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::state::CommandVisitor**

Interface for an Object that can visit the various Command Objects that are sent from and to this client.

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**
- namespace **activemq::state**

7.105 src/main/activemq/state/CommandVisitorAdapter.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/state/CommandVisitor.h>
#include <activemq/core/ActiveMQConstants.h>
#include <activemq/commands/ConnectionInfo.h>
#include <activemq/commands/SessionInfo.h>
#include <activemq/commands/ProducerInfo.h>
#include <activemq/commands/ConsumerInfo.h>
#include <activemq/commands/ConnectionId.h>
#include <activemq/commands/SessionId.h>
#include <activemq/commands/ProducerId.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/DestinationInfo.h>
#include <activemq/commands/RemoveSubscriptionInfo.h>
#include <activemq/commands/Message.h>
#include <activemq/commands/MessageAck.h>
```

```
#include <activemq/commands/MessagePull.h>
#include <activemq/commands/TransactionInfo.h>
#include <activemq/commands/WireFormatInfo.h>
#include <activemq/commands/ProducerAck.h>
#include <activemq/commands/MessageDispatch.h>
#include <activemq/commands/MessageDispatchNotification.h>
#include <activemq/commands/ControlCommand.h>
#include <activemq/commands/ConnectionError.h>
#include <activemq/commands/ConnectionControl.h>
#include <activemq/commands/ConsumerControl.h>
#include <activemq/commands/ShutdownInfo.h>
#include <activemq/commands/KeepAliveInfo.h>
#include <activemq/commands/FlushCommand.h>
#include <activemq/commands/BrokerError.h>
#include <activemq/commands/BrokerInfo.h>
#include <activemq/commands/RemoveInfo.h>
#include <activemq/commands/ReplayCommand.h>
#include <activemq/commands/Response.h>
```

Data Structures

- class **activemq::state::CommandVisitorAdapter**

*Default Implementation of a **CommandVisitor** (p. 996) that returns NULL for all calls.*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::state**

7.106 src/main/activemq/state/ConnectionState.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ConnectionInfo.h>
#include <activemq/commands/DestinationInfo.h>
#include <activemq/commands/SessionInfo.h>
```

```
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/ProducerId.h>
#include <activemq/commands/TransactionId.h>
#include <activemq/state/ConsumerState.h>
#include <activemq/state/ProducerState.h>
#include <activemq/state/SessionState.h>
#include <activemq/state/TransactionState.h>
#include <decaf/util/concurrent/atomic/AtomicBoolean.h>
#include <decaf/util/concurrent/ConcurrentStlMap.h>
#include <decaf/util/StlList.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <memory>
```

Data Structures

- class `activemq::state::ConnectionState`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::state`

7.107 src/main/activemq/state/ConnectionStateTracker.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ConnectionId.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/state/CommandVisitorAdapter.h>
#include <activemq/state/ConnectionState.h>
#include <activemq/state/ConsumerState.h>
#include <activemq/state/ProducerState.h>
#include <activemq/state/SessionState.h>
#include <activemq/state/TransactionState.h>
#include <activemq/state/Tracked.h>
#include <activemq/transport/Transport.h>
```

```
#include <decaf/util/concurrent/ConcurrentStlMap.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::state::ConnectionStateTracker**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::state**

7.108 src/main/activemq/state/ConsumerState.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ConsumerInfo.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <memory>
```

Data Structures

- class **activemq::state::ConsumerState**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::state**

7.109 src/main/activemq/state/ProducerState.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ProducerInfo.h>
#include <decaf/lang/Pointer.h>
#include <string>
```

```
#include <memory>
```

Data Structures

- class `activemq::state::ProducerState`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::state`

7.110 `src/main/activemq/state/SessionState.h` File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/SessionInfo.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/ProducerId.h>
#include <activemq/state/ConsumerState.h>
#include <activemq/state/ProducerState.h>
#include <decaf/util/concurrent/atomic/AtomicBoolean.h>
#include <decaf/util/concurrent/ConcurrentStlMap.h>
#include <string>
#include <memory>
```

Data Structures

- class `activemq::state::SessionState`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::state`

7.111 `src/main/activemq/state/Tracked.h` File Reference

```
#include <activemq/util/Config.h>
```

```
#include <activemq/commands/Response.h>
#include <decaf/lang/Runnable.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class `activemq::state::Tracked`

Namespaces

- namespace `activemq`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `activemq::state`

7.112 src/main/activemq/state/TransactionState.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/Command.h>
#include <activemq/commands/TransactionId.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/StlList.h>
#include <decaf/util/concurrent/atomic/AtomicBoolean.h>
#include <string>
#include <memory>
```

Data Structures

- class `activemq::state::TransactionState`

Namespaces

- namespace `activemq`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `activemq::state`

7.113 src/main/activemq/threads/CompositeTask.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/threads/Task.h>
```

Data Structures

- class **activemq::threads::CompositeTask**
Represents a single task that can be part of a set of Tasks that are contained in a CompositeTaskRunner (p. 1017).

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::threads**

7.114 src/main/activemq/threads/CompositeTaskRunner.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/threads/TaskRunner.h>
#include <activemq/threads/CompositeTask.h>
#include <decaf/util/StlSet.h>
#include <decaf/util/StlList.h>
#include <decaf/lang/Thread.h>
#include <decaf/lang/Runnable.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::threads::CompositeTaskRunner**
A Task (p. 2956) Runner that can contain one or more CompositeTasks that are each checked for pending work and run if any is present in the order that the tasks were added.

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::threads**

7.115 src/main/activemq/threads/DedicatedTaskRunner.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/threads/TaskRunner.h>
#include <activemq/threads/Task.h>
#include <decaf/lang/Thread.h>
#include <decaf/lang/Runnable.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::threads::DedicatedTaskRunner**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::threads**

7.116 src/main/activemq/threads/Task.h File Reference

```
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::threads::Task**

Represents a unit of work that requires one or more iterations to complete.

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::threads**

7.117 src/main/activemq/threads/TaskRunner.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/threads/Task.h>
```

Data Structures

- class **activemq::threads::TaskRunner**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::threads**

7.118 src/main/activemq/transport/AbstractTransportFactory.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/Transport.h>
#include <activemq/transport/TransportFactory.h>
#include <activemq/wireformat/WireFormat.h>
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/Properties.h>
```

Data Structures

- class **activemq::transport::AbstractTransportFactory**
*Abstract implementation of the **TransportFactory** (p. 3071) interface, providing the base functionality that's common to most of the **TransportFactory** (p. 3071) instances.*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::transport**

7.119 src/main/activemq/transport/CompositeTransport.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/Transport.h>
#include <decaf/net/URI.h>
#include <decaf/util/List.h>
```

Data Structures

- class **activemq::transport::CompositeTransport**
*A Composite **Transport** (p. 3066) is a **Transport** (p. 3066) implementation that is composed of several **Transports**.*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::transport**

7.120 src/main/activemq/transport/correlator/FutureResponse.h File Reference

```
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/concurrent/CountDownLatch.h>
#include <activemq/commands/Response.h>
#include <activemq/exceptions/ActiveMQException.h>
```

Data Structures

- class **activemq::transport::correlator::FutureResponse**
A container that holds a response object.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::transport**
- namespace **activemq::transport::correlator**

7.121 src/main/activemq/transport/correlator/ResponseCorrelator.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/TransportFilter.h>
#include <activemq/transport/correlator/FutureResponse.h>
#include <activemq/commands/Command.h>
#include <activemq/commands/Response.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/concurrent/Concurrent.h>
#include <decaf/util/concurrent/atomic/AtomicInteger.h>
#include <map>
#include <stdio.h>
```

Data Structures

- class **activemq::transport::correlator::ResponseCorrelator**

This type of transport filter is responsible for correlating asynchronous responses with requests.

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::transport**
- namespace **activemq::transport::correlator**

7.122 src/main/activemq/transport/DefaultTransportListener.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/TransportListener.h>
#include <activemq/commands/Command.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class `activemq::transport::DefaultTransportListener`

Namespaces

- namespace `activemq`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `activemq::transport`

7.123 src/main/activemq/transport/failover/BackupTransport.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/Transport.h>
#include <activemq/transport/DefaultTransportListener.h>
#include <decaf/net/URI.h>
#include <decaf/lang/Pointer.h>
#include <memory>
```

Data Structures

- class `activemq::transport::failover::BackupTransport`

Namespaces

- namespace `activemq`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `activemq::transport`
- namespace `activemq::transport::failover`

7.124 src/main/activemq/transport/failover/BackupTransportPool.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/threads/CompositeTask.h>
#include <activemq/threads/CompositeTaskRunner.h>
#include <activemq/transport/failover/CloseTransportsTask.h>
#include <activemq/transport/failover/BackupTransport.h>
```

```
#include <activemq/transport/failover/URIPool.h>
#include <decaf/lang/Pointer.h>
#include <decaf/io/IOException.h>
#include <decaf/util/StlList.h>
```

Data Structures

- class **activemq::transport::failover::BackupTransportPool**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::transport**
- namespace **activemq::transport::failover**

7.125 src/main/activemq/transport/failover/CloseTransportsTask.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/threads/CompositeTask.h>
#include <activemq/transport/Transport.h>
#include <decaf/util/StlQueue.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::transport::failover::CloseTransportsTask**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::transport**
- namespace **activemq::transport::failover**

7.126 src/main/activemq/transport/failover/FailoverTransport.h File Reference

```
#include <activemq/util/Config.h>
```

```
#include <activemq/commands/Command.h>
#include <activemq/threads/TaskRunner.h>
#include <activemq/threads/CompositeTaskRunner.h>
#include <activemq/state/ConnectionStateTracker.h>
#include <activemq/transport/CompositeTransport.h>
#include <activemq/transport/failover/BackupTransportPool.h>
#include <activemq/transport/failover/CloseTransportsTask.h>
#include <activemq/transport/failover/FailoverTransportListener.h>
#include <activemq/transport/failover/URIPool.h>
#include <activemq/wireformat/WireFormat.h>
#include <decaf/util/StlList.h>
#include <decaf/util/StlMap.h>
#include <decaf/util/Properties.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/concurrent/atomic/AtomicReference.h>
#include <decaf/net/URI.h>
#include <decaf/io/IOException.h>
```

Data Structures

- class `activemq::transport::failover::FailoverTransport`

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::transport`
- namespace `activemq::transport::failover`

7.127 src/main/activemq/transport/failover/FailoverTransportFactory.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/AbstractTransportFactory.h>
#include <activemq/transport/Transport.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/wireformat/WireFormat.h>
#include <decaf/net/URI.h>
```

```
#include <decaf/util/Properties.h>
```

Data Structures

- class **activemq::transport::failover::FailoverTransportFactory**
*Creates an instance of a **FailoverTransport** (p. 1512).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::transport**
- namespace **activemq::transport::failover**

7.128 src/main/activemq/transport/failover/FailoverTransportListener. File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/TransportListener.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::transport::failover::FailoverTransportListener**
*Utility class used by the **Transport** (p. 3066) to perform the work of responding to events from the active **Transport** (p. 3066).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::transport**
- namespace **activemq::transport::failover**

7.129 src/main/activemq/transport/failover/URIPool.h File Reference

```
#include <activemq/util/Config.h>
#include <decaf/net/URI.h>
```


7.130 src/main/activemq/transport/inactivity/InactivityMonitor.h File Reference 297

```
#include <decaf/util/StlList.h>
#include <decaf/lang/exceptions/NoSuchElementException.h>
```

Data Structures

- class **activemq::transport::failover::URIPool**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::transport**
- namespace **activemq::transport::failover**

7.130 src/main/activemq/transport/inactivity/InactivityMonitor.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/TransportFilter.h>
#include <activemq/commands/Command.h>
#include <activemq/commands/Response.h>
#include <activemq/commands/WireFormatInfo.h>
#include <activemq/wireformat/WireFormat.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/Timer.h>
#include <decaf/util/Properties.h>
#include <decaf/util/concurrent/atomic/AtomicBoolean.h>
```

Data Structures

- class **activemq::transport::inactivity::InactivityMonitor**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::transport**
- namespace **activemq::transport::inactivity**

7.131 src/main/activemq/transport/inactivity/ReadChecker.h File Reference

```
#include <activemq/util/Config.h>
#include <decaf/util/TimerTask.h>
```

Data Structures

- class **activemq::transport::inactivity::ReadChecker**

Runnable class that is used by the {}.

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::transport**
- namespace **activemq::transport::inactivity**

7.132 src/main/activemq/transport/inactivity/WriteChecker.h File Reference

```
#include <activemq/util/Config.h>
#include <decaf/util/TimerTask.h>
```

Data Structures

- class **activemq::transport::inactivity::WriteChecker**

Runnable class used by the {}.

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::transport**
- namespace **activemq::transport::inactivity**

7.133 src/main/activemq/transport/IOTransport.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/Transport.h>
#include <activemq/transport/TransportListener.h>
#include <activemq/commands/Command.h>
#include <activemq/commands/Response.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/wireformat/WireFormat.h>
#include <decaf/lang/Runnable.h>
#include <decaf/lang/Thread.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/util/logging/LoggerDefines.h>
#include <memory>
```

Data Structures

- class **activemq::transport::IOTransport**

*Implementation of the **Transport** (p. 3066) interface that performs marshaling of commands to IO streams.*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::transport**

7.134 src/main/activemq/transport/logging/LoggingTransport.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/TransportFilter.h>
#include <decaf/util/logging/LoggerDefines.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::transport::logging::LoggingTransport**
A transport filter that logs commands as they are sent/received.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::transport**
- namespace **activemq::transport::logging**

7.135 src/main/activemq/transport/mock/InternalCommandListener.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/mock/ResponseBuilder.h>
#include <activemq/transport/DefaultTransportListener.h>
#include <decaf/lang/Thread.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/StlQueue.h>
#include <decaf/util/concurrent/Concurrent.h>
#include <decaf/util/concurrent/atomic/AtomicInteger.h>
#include <decaf/util/concurrent/CountDownLatch.h>
```

Data Structures

- class **activemq::transport::mock::InternalCommandListener**
*Listens for Commands sent from the **MockTransport** (p. 2227).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::transport**
- namespace **activemq::transport::mock**

7.136 src/main/activemq/transport/mock/MockTransport.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/transport/Transport.h>
#include <activemq/transport/TransportListener.h>
#include <activemq/transport/DefaultTransportListener.h>
#include <activemq/transport/mock/ResponseBuilder.h>
#include <activemq/transport/mock/InternalCommandListener.h>
#include <activemq/wireformat/WireFormat.h>
#include <decaf/lang/Thread.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/StlQueue.h>
#include <decaf/util/concurrent/Concurrent.h>
#include <decaf/util/concurrent/atomic/AtomicInteger.h>
#include <decaf/util/concurrent/CountDownLatch.h>
#include <cms/Message.h>
#include <map>
#include <set>
```

Data Structures

- class **activemq::transport::mock::MockTransport**
*The **MockTransport** (p. 2227) defines a base level **Transport** (p. 3066) class that is intended to be used in place of an a regular protocol **Transport** (p. 3066) such as TCP.*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::transport**
- namespace **activemq::transport::mock**

7.137 src/main/activemq/transport/mock/MockTransportFactory.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/AbstractTransportFactory.h>
```

Data Structures

- class **activemq::transport::mock::MockTransportFactory**

Manufactures MockTransports, which are objects that read from input streams and write to output streams.

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::transport**
- namespace **activemq::transport::mock**

7.138 src/main/activemq/transport/mock/ResponseBuilder.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/Command.h>
#include <activemq/commands/Response.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/StlQueue.h>
```

Data Structures

- class **activemq::transport::mock::ResponseBuilder**

Interface for all Protocols to implement that defines the behavior of the Broker in response to messages of that protocol.

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::transport**
- namespace **activemq::transport::mock**

7.139 src/main/activemq/transport/tcp/TcpTransport.h File Reference

```
#include <activemq/io/LoggingInputStream.h>
```

```
#include <activemq/io/LoggingOutputStream.h>
#include <activemq/util/Config.h>
#include <activemq/transport/TransportFilter.h>
#include <decaf/net/Socket.h>
#include <decaf/net/URI.h>
#include <decaf/util/Properties.h>
#include <decaf/lang/Pointer.h>
#include <decaf/io/BufferedInputStream.h>
#include <decaf/io/BufferedOutputStream.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <memory>
```

Data Structures

- class **activemq::transport::tcp::TcpTransport**

*Implements a TCP/IP based transport filter, this transport is meant to wrap an instance of an **IOTransport** (p. 1709).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::transport**
- namespace **activemq::transport::tcp**

7.140 src/main/activemq/transport/tcp/TcpTransportFactory.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/AbstractTransportFactory.h>
#include <activemq/exceptions/ActiveMQException.h>
```

Data Structures

- class **activemq::transport::tcp::TcpTransportFactory**

*Factory Responsible for creating the **TcpTransport** (p. 2967).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::transport**
- namespace **activemq::transport::tcp**

7.141 src/main/activemq/transport/Transport.h File Reference

```
#include <decaf/io/InputStream.h>
#include <decaf/io/OutputStream.h>
#include <decaf/io/IOException.h>
#include <decaf/io/Closeable.h>
#include <decaf/net/URI.h>
#include <decaf/lang/Pointer.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/Command.h>
#include <activemq/commands/Response.h>
#include <typeinfo>
```

Data Structures

- class **activemq::transport::Transport**

Interface for a transport layer for command objects.

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::transport**

7.142 src/main/activemq/transport/TransportFactory.h File Reference

```
#include <activemq/util/Config.h>
```



```
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/transport/Transport.h>
#include <decaf/net/URI.h>
#include <decaf/util/Properties.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::transport::TransportFactory**
Defines the interface for Factories that create Transports or TransportFilters.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::transport**

7.143 src/main/activemq/transport/TransportFilter.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/transport/Transport.h>
#include <activemq/commands/Command.h>
#include <activemq/transport/TransportListener.h>
#include <decaf/lang/Pointer.h>
#include <typeinfo>
```

Data Structures

- class **activemq::transport::TransportFilter**
A filter on the transport layer.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::transport**

7.144 src/main/activemq/transport/TransportListener.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/Command.h>
#include <decaf/lang/Exception.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::transport::TransportListener**

A listener of asynchronous exceptions from a command transport object.

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::transport**

7.145 src/main/activemq/transport/TransportRegistry.h File Reference

```
#include <activemq/util/Config.h>
#include <string>
#include <vector>
#include <activemq/transport/TransportFactory.h>
#include <decaf/util/StlMap.h>
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
```

Data Structures

- class **activemq::transport::TransportRegistry**

*Registry of all **Transport** (p. 3066) **Factories** that are available to the client at runtime.*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::transport**

7.146 src/main/activemq/util/ActiveMQProperties.h File Reference

```
#include <map>
#include <string>
#include <sstream>
#include <activemq/util/Config.h>
#include <cms/CMSProperties.h>
#include <decaf/util/Properties.h>
```

Data Structures

- class **activemq::util::ActiveMQProperties**
Implementation of the CMSProperties interface that delegates to a decaf::util::Properties (p. 2494) object.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::util**

7.147 src/main/activemq/util/CMSExceptionSupport.h File Reference

```
#include <activemq/util/Config.h>
#include <cms/CMSException.h>
#include <cms/CMSSecurityException.h>
#include <cms/MessageEOFException.h>
#include <cms/MessageFormatException.h>
#include <cms/MessageNotReadableException.h>
#include <cms/MessageNotWritableException.h>
#include <cms/InvalidClientIdException.h>
#include <cms/InvalidDestinationException.h>
```

```
#include <cms/InvalidSelectorException.h>
#include <cms/IllegalStateException.h>
#include <cms/UnsupportedOperationException.h>
#include <decaf/lang/Exception.h>
#include <string>
```

Data Structures

- class **activemq::util::CMSExceptionSupport**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::util**

Defines

- **#define AMQ_CATCH_ALL_THROW_CMSEXCEPTION()**

Macro for catching an exception of one type and then re-throwing as a Basic CMSException, good for cases where the method isn't specific about what CMS Exceptions are thrown, bad if you need to throw an exception of MessageNotReadableException for instance.

7.147.1 Define Documentation

7.147.1.1 **#define AMQ_CATCH_ALL_THROW_CMSEXCEPTION()**

Macro for catching an exception of one type and then re-throwing as a Basic CMSException, good for cases where the method isn't specific about what CMS Exceptions are thrown, bad if you need to throw an exception of MessageNotReadableException for instance.

Referenced by `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::acknowledge()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::clearBody()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::clearProperties()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::equals()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getBooleanProperty()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getBytesProperty()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getCMSMessageID()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getDoubleProperty()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getFloatProperty()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getIntProperty()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getLongProperty()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getPropertyNames()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getShortProperty()`, `activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getStringProperty()`, `activemq::commands::ActiveMQMessageTemplate<`

```

cms::ObjectMessage >::propertyExists(), activemq::commands::ActiveMQMessageTemplate<
cms::ObjectMessage >::setBooleanProperty(), activemq::commands::ActiveMQMessageTemplate<
cms::ObjectMessage >::setByteProperty(), activemq::commands::ActiveMQMessageTemplate<
cms::ObjectMessage >::setCMSDestination(), activemq::commands::ActiveMQMessageTemplate<
cms::ObjectMessage >::setCMSReplyTo(), activemq::commands::ActiveMQMessageTemplate<
cms::ObjectMessage >::setDoubleProperty(), activemq::commands::ActiveMQMessageTemplate<
cms::ObjectMessage >::setFloatProperty(), activemq::commands::ActiveMQMessageTemplate<
cms::ObjectMessage >::setIntProperty(), activemq::commands::ActiveMQMessageTemplate<
cms::ObjectMessage >::setLongProperty(), activemq::commands::ActiveMQMessageTemplate<
cms::ObjectMessage >::setShortProperty(), and activemq::commands::ActiveMQMessageTemplate<
cms::ObjectMessage >::setStringProperty().

```

7.148 src/main/activemq/util/CompositeData.h File Reference

```

#include <activemq/util/Config.h>
#include <decaf/util/Properties.h>
#include <decaf/util/StlList.h>
#include <decaf/net/URI.h>
#include <decaf/net/URISyntaxException.h>

```

Data Structures

- class **activemq::util::CompositeData**
Represents a Composite URI.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::util**

7.149 src/main/activemq/util/Config.h File Reference

Defines

- #define **AMQCPP_API**
- #define **HAVE_UUID_UUID_H**
- #define **HAVE_UUID_T**
- #define **HAVE_PTHREAD_H**

7.149.1 Define Documentation

7.149.1.1 `#define AMQCPP_API`

7.149.1.2 `#define HAVE_PTHREAD_H`

7.149.1.3 `#define HAVE_UUID_T`

7.149.1.4 `#define HAVE_UUID_UUID_H`

7.150 src/main/cms/Config.h File Reference

Defines

- `#define CMS_API`

7.150.1 Define Documentation

7.150.1.1 `#define CMS_API`

7.151 src/main/decaf/util/Config.h File Reference

Defines

- `#define DECAF_API`
- `#define HAVE_UUID_UUID_H`
- `#define HAVE_UUID_T`
- `#define HAVE_PTHREAD_H`
- `#define DECAF_UNUSED`

7.151.1 Define Documentation

7.151.1.1 `#define DECAF_API`

7.151.1.2 `#define DECAF_UNUSED`

7.151.1.3 `#define HAVE_PTHREAD_H`

7.151.1.4 `#define HAVE_UUID_T`

7.151.1.5 `#define HAVE_UUID_UUID_H`

7.152 src/main/activemq/util/LongSequenceGenerator.h File Reference

```
#include <activemq/util/Config.h>
```

```
#include <decaf/util/concurrent/Mutex.h>
```

Data Structures

- class **activemq::util::LongSequenceGenerator**

This class is used to generate a sequence of long long values that are incremented each time a new value is requested.

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::util**

7.153 src/main/activemq/util/MemoryUsage.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/util/Usage.h>
#include <decaf/util/concurrent/Mutex.h>
```

Data Structures

- class **activemq::util::MemoryUsage**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::util**

7.154 src/main/activemq/util/PrimitiveList.h File Reference

```
#include <string>
#include <vector>
#include <decaf/util/StlList.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <stdio.h>
#include <activemq/util/PrimitiveValueNode.h>
```

```
#include <activemq/util/PrimitiveValueConverter.h>
```

Data Structures

- class **activemq::util::PrimitiveList**

List of primitives.

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::util**

7.155 src/main/activemq/util/PrimitiveMap.h File Reference

```
#include <string>
#include <vector>
#include <activemq/util/Config.h>
#include <decaf/util/Config.h>
#include <decaf/util/StlMap.h>
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <activemq/util/PrimitiveValueNode.h>
#include <activemq/util/PrimitiveValueConverter.h>
```

Data Structures

- class **activemq::util::PrimitiveMap**

Map of named primitives.

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::util**

7.156 src/main/activemq/util/PrimitiveValueConverter.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/util/PrimitiveValueNode.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <string>
```

Data Structures

- class **activemq::util::PrimitiveValueConverter**
*Class controls the conversion of data contained in a **PrimitiveValueNode** (p. 2398) from one type to another.*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::util**

7.157 src/main/activemq/util/PrimitiveValueNode.h File Reference

```
#include <activemq/util/Config.h>
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/util/Map.h>
#include <decaf/util/List.h>
```

Data Structures

- class **activemq::util::PrimitiveValueNode**
Class that wraps around a single value of one of the many types.
- union **activemq::util::PrimitiveValueNode::PrimitiveValue**
Define a union type comprised of the various types.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::util**

7.158 src/main/activemq/util/URISupport.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/util/CompositeData.h>
#include <decaf/util/Properties.h>
#include <decaf/util/StlList.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
```

Data Structures

- class **activemq::util::URISupport**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::util**

7.159 src/main/activemq/util/Usage.h File Reference

```
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::util::Usage**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::util**

7.160 src/main/activemq/wireformat/MarshalAware.h File Reference

```
#include <vector>
```

7.161

src/main/activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h

File Reference

3315

```
#include <decaf/io/IOException.h>
```

```
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::wireformat::MarshalAware**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**

7.161 src/main/activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/DataStreamMarshaller.h>
```

```
#include <activemq/wireformat/openwire/utls/HexTable.h>
```

```
#include <activemq/commands/MessageId.h>
```

```
#include <activemq/commands/ProducerId.h>
```

```
#include <activemq/commands/TransactionId.h>
```

```
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller**

Base class for all Marshallers that marshal DataStructures to and from the wire using the Open-Wire protocol.

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**

- namespace **activemq::wireformat::openwire**

- namespace **activemq::wireformat::openwire::marshal**

7.162 src/main/activemq/wireformat/openwire/marshal/DataStreamM File Reference

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::DataStreamMarshaller**

Base class for all classes that marshal commands for Openwire.

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**

7.163 src/main/activemq/wireformat/openwire/marshal/PrimitiveType File Reference

```
#include <cms/CMSException.h>
#include <activemq/util/Config.h>
#include <activemq/util/PrimitiveValueNode.h>
#include <activemq/util/PrimitiveMap.h>
#include <activemq/util/PrimitiveList.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/IOException.h>
#include <string>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller**

This class wraps the functionality needed to marshal a primitive map to the Openwire Format's expectation of what the map looks like on the wire.

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**

7.164 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQ File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 150).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.165 src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQBlobMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 162).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.166 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQBlobMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 146).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.167 src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQ File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 154).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.168 src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQBlobMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 158).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.169 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQBlobMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```


Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 186).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.170 src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQBytesMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 198).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.171 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQBytesMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 182).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.172 src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQBytesMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 190).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.173 src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQBytesMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 194).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.174 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQDestinationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 254).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.175 src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQDestinationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 265).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.176 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQDestinationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 250).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.177 src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQDestinationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 258).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.178 src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQDestinationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 262).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.179 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQMapMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 288).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.180 src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQMapMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 300).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.181 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQMapMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```


Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 284).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.182 src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQMapMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 292).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.183 src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQMapMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 296).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.184 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQMapMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 311).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.185 src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 323).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.186 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 307).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.187 src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 315).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.188 src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQ File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 319).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.189 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQObjectMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 351).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.190 src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQObjectMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

- class `activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller`

*Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 363).*

Namespaces

- namespace `activemq`
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace `activemq::wireformat`
- namespace `activemq::wireformat::openwire`
- namespace `activemq::wireformat::openwire::marshal`
- namespace `activemq::wireformat::openwire::marshal::v2`

7.191 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQObjectMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/MessageMarshaller.h>  
#include <decaf/io/DataInputStream.h>  
#include <decaf/io/DataOutputStream.h>  
#include <decaf/io/IOException.h>  
#include <activemq/util/Config.h>  
#include <activemq/commands/DataStructure.h>  
#include <activemq/wireformat/openwire/OpenWireFormat.h>  
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class `activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller`

*Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 347).*

Namespaces

- namespace `activemq`
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace `activemq::wireformat`
- namespace `activemq::wireformat::openwire`
- namespace `activemq::wireformat::openwire::marshal`
- namespace `activemq::wireformat::openwire::marshal::v3`

7.192 src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQObjectMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 355).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.193 src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQObjectMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```


7.194

src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQQueueMarshaller.h
File Reference 3337
Data Structures

- class `activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller`

*Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 359).*

Namespaces

- namespace `activemq`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `activemq::wireformat`
- namespace `activemq::wireformat::openwire`
- namespace `activemq::wireformat::openwire::marshal`
- namespace `activemq::wireformat::openwire::marshal::v5`

7.194 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQQueueMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class `activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller`

*Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 386).*

Namespaces

- namespace `activemq`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `activemq::wireformat`
- namespace `activemq::wireformat::openwire`
- namespace `activemq::wireformat::openwire::marshal`
- namespace `activemq::wireformat::openwire::marshal::v1`

7.195 src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQQueueMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 398).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.196 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQQueueMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

7.197

src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQQueueMarshaller.h

File Reference

3339

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller**

Marshaling code for Open Wire Format for ActiveMQQueueMarshaller (p. 382).

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.197 src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQQueueMarshaller.h

File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller**

Marshaling code for Open Wire Format for ActiveMQQueueMarshaller (p. 390).

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.198 src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQQueueMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 394).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.199 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQQueueMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 439).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.200 src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQS File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 451).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.201 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQS File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 436).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.202 src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQS File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

- class **activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 443).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.203 src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQS File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 447).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.204 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTempDestinationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 462).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.205 src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTempDestinationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```


- class **activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller**
(p. 473).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.206 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTempDestinationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller**
(p. 459).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**

- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.207 src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTempDestinationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 466).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.208 src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTempDestinationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
```

7.209 src/main/activemq/wireformat/openwire/marsh shal/v1/ActiveMQTempQueueMarshaller.h File Reference

3347

```
#include <activemq/commands/DataStructure.h>
```

```
#include <activemq/wireformat/openwire/OpenWireFormat.h>
```

```
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 470).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.209 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTempQueueMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/ActiveMQTempDestinationMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
```

```
#include <decaf/io/DataOutputStream.h>
```

```
#include <decaf/io/IOException.h>
```

```
#include <activemq/util/Config.h>
```

```
#include <activemq/commands/DataStructure.h>
```

```
#include <activemq/wireformat/openwire/OpenWireFormat.h>
```

```
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 485).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.210 src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTempDestinationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/ActiveMQTempDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 497).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.211 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTempDestinationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/ActiveMQTempDestinationMarshaller.h>
```

7.212 src/main/activemq/wireformat/openwire/marsh- shal/v4/ActiveMQTempQueueMarshaller.h File

Reference

3349

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 481).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.212 src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTempQueueMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/ActiveMQTempDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 489).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.213 src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTempDestinationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/ActiveMQTempDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 493).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.214 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTempDestinationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/ActiveMQTempDestinationMarshaller.h>
```

7.215 src/main/activemq/wireformat/openwire/marsh- shal/v2/ActiveMQTempTopicMarshaller.h File

Reference

3351

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 509).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.215 src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTempTopicMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/ActiveMQTempDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 521).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.216 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTempDestinationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/ActiveMQTempDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 505).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.217 src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTempDestinationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/ActiveMQTempDestinationMarshaller.h>
```


7.218 src/main/activemq/wireformat/openwire/marsh shal/v5/ActiveMQTempTopicMarshaller.h File

Reference

3353

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 513).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.218 src/main/activemq/wireformat/openwire/marsh shal/v5/ActiveMQTempTopicMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marsh/v5/ActiveMQTempDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 517).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.219 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTextMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 533).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.220 src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTextMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/MessageMarshaller.h>
```

7.221 src/main/activemq/wireformat/openwire/marsh- shal/v3/ActiveMQTextMessageMarshaller.h File

Reference

3355

```
#include <decaf/io/DataInputStream.h>

#include <decaf/io/DataOutputStream.h>

#include <decaf/io/IOException.h>

#include <activemq/util/Config.h>

#include <activemq/commands/DataStructure.h>

#include <activemq/wireformat/openwire/OpenWireFormat.h>

#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 545).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.221 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTextMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/MessageMarshaller.h>

#include <decaf/io/DataInputStream.h>

#include <decaf/io/DataOutputStream.h>

#include <decaf/io/IOException.h>

#include <activemq/util/Config.h>

#include <activemq/commands/DataStructure.h>

#include <activemq/wireformat/openwire/OpenWireFormat.h>

#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 529).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.222 src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTextMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 537).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.223 src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTextMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/MessageMarshaller.h>
```

7.224

src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTopicMarshaller.h
File Reference 3357

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataSetStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 541).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.224 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTopicMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataSetStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller**

*Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 557).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.225 src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTopicMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 569).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.226 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTopicMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/ActiveMQDestinationMarshaller.h>
```

7.227

src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTopicMarshaller.h
File Reference 3359

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 553).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.227 src/main/activemq/wireformat/openwire/marshal/v4/ActiveMQTopicMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 561).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.228 src/main/activemq/wireformat/openwire/marshal/v5/ActiveMQTopicMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller**
*Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 565).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.229 src/main/activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
```


7.230

src/main/activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h
File Reference 3361

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataSetStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller**
*Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 610).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.230 src/main/activemq/wireformat/openwire/marshal/v2/BaseComm File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataSetStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller**
*Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 630).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.231 src/main/activemq/wireformat/openwire/marshal/v3/BaseComm File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller**
*Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 603).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.232 src/main/activemq/wireformat/openwire/marshal/v4/BaseComm File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
```

7.233

src/main/activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h
File Reference 3363

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataSetStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller**
*Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 616).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.233 src/main/activemq/wireformat/openwire/marshal/v5/BaseComm File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataSetStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller**
*Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 623).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.234 src/main/activemq/wireformat/openwire/marshal/v1/BrokerIdM File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller**
*Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 698).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.235 src/main/activemq/wireformat/openwire/marshal/v2/BrokerIdM File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataSetStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller**

*Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 709).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.236 src/main/activemq/wireformat/openwire/marshal/v3/BrokerIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataSetStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller**

*Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 694).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.237 src/main/activemq/wireformat/openwire/marshal/v4/BrokerIdM File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller**
*Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 701).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.238 src/main/activemq/wireformat/openwire/marshal/v5/BrokerIdM File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataSetStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller**

*Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 705).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.239 src/main/activemq/wireformat/openwire/marshal/v1/BrokerInfo File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataSetStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller**

*Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 725).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.240 src/main/activemq/wireformat/openwire/marshal/v2/BrokerInfo File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller**
*Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 737).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.241 src/main/activemq/wireformat/openwire/marshal/v3/BrokerInfo File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
```



```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataSetStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller**

*Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 721).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.242 src/main/activemq/wireformat/openwire/marshal/v4/BrokerInfo File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataSetStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller**

*Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 729).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.243 src/main/activemq/wireformat/openwire/marshal/v5/BrokerInfo File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller**

*Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 733).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.244 src/main/activemq/wireformat/openwire/marshal/v1/Connection File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
```

7.245 src/main/activemq/wireformat/openwire/marsh shal/v2/ConnectionControlMarshaller.h File

Reference

3371

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller**

*Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1063).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.245 src/main/activemq/wireformat/openwire/marsh shal/v2/Connection File Reference

```
#include <activemq/wireformat/openwire/marsh/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller**

*Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1075).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.246 src/main/activemq/wireformat/openwire/marshal/v3/ConnectionControlMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller**

*Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1059).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.247 src/main/activemq/wireformat/openwire/marshal/v4/ConnectionControlMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
```

7.248 src/main/activemq/wireformat/openwire/marsh shal/v5/ConnectionControlMarshaller.h File

Reference

3373

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller**

*Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1067).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.248 src/main/activemq/wireformat/openwire/marsh shal/v5/Connection File Reference

```
#include <activemq/wireformat/openwire/marsh/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller**

*Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 1071).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.249 src/main/activemq/wireformat/openwire/marshal/v1/Connection File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller**
*Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1087).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.250 src/main/activemq/wireformat/openwire/marshal/v2/Connection File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
```

7.251

src/main/activemq/wireformat/openwire/marshal/v3/ConnectionErrorMarshaller.h
File Reference 3375

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataSetStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller**
*Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1099).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.251 src/main/activemq/wireformat/openwire/marshal/v3/ConnectionErrorMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataSetStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller**
*Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1083).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.252 src/main/activemq/wireformat/openwire/marshal/v4/Connection File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller**
*Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1091).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.253 src/main/activemq/wireformat/openwire/marshal/v5/Connection File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
```


7.254

src/main/activemq/wireformat/openwire/marshal/v1/ConnectionIdMarshaller.h

File Reference

3377

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller**
*Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 1095).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.254 src/main/activemq/wireformat/openwire/marshal/v1/ConnectionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller**
*Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1113).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.255 src/main/activemq/wireformat/openwire/marshal/v2/ConnectionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller**
*Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1124).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.256 src/main/activemq/wireformat/openwire/marshal/v3/ConnectionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
```

7.257

src/main/activemq/wireformat/openwire/marshal/v4/ConnectionIdMarshaller.h

File Reference

3379

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller**

*Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1109).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.257 src/main/activemq/wireformat/openwire/marshal/v4/Connection

File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller**

*Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1116).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.258 src/main/activemq/wireformat/openwire/marshal/v5/Connection File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller**
*Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 1120).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.259 src/main/activemq/wireformat/openwire/marshal/v1/Connection File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
```

7.260

src/main/activemq/wireformat/openwire/marshal/v2/ConnectionInfoMarshaller.h
File Reference

3381

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataSetStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller**
*Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1142).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.260 src/main/activemq/wireformat/openwire/marshal/v2/ConnectionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataSetStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller**
*Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1150).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.261 src/main/activemq/wireformat/openwire/marshal/v3/Connection File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller**
*Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1134).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.262 src/main/activemq/wireformat/openwire/marshal/v4/Connection File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
```

7.263

src/main/activemq/wireformat/openwire/marshal/v5/ConnectionInfoMarshaller.h
File Reference

3383

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataSetStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller**
*Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1138).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.263 src/main/activemq/wireformat/openwire/marshal/v5/ConnectionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataSetStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller**
*Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1146).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.264 src/main/activemq/wireformat/openwire/marshal/v1/ConsumerC File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller**

*Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1178).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.265 src/main/activemq/wireformat/openwire/marshal/v2/ConsumerC File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
```


7.266 src/main/activemq/wireformat/openwire/marsh shal/v3/ConsumerControlMarshaller.h File

Reference

3385

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller**

*Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1186).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.266 src/main/activemq/wireformat/openwire/marsh shal/v3/ConsumerC File Reference

```
#include <activemq/wireformat/openwire/marsh/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller**

*Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1170).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.267 src/main/activemq/wireformat/openwire/marshal/v4/ConsumerC File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller**

*Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1174).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.268 src/main/activemq/wireformat/openwire/marshal/v5/ConsumerC File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
```

7.269

src/main/activemq/wireformat/openwire/marshal/v1/ConsumerIdMarshaller.h File Reference 3387

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller**

*Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1182).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.269 src/main/activemq/wireformat/openwire/marshal/v1/ConsumerIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller**

*Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1202).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.270 src/main/activemq/wireformat/openwire/marshal/v2/ConsumerIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller**
*Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1210).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.271 src/main/activemq/wireformat/openwire/marshal/v3/ConsumerIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
```

7.272

src/main/activemq/wireformat/openwire/marshal/v4/ConsumerIdMarshaller.h File Reference 3389

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataSetStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller**

*Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1195).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.272 src/main/activemq/wireformat/openwire/marshal/v4/ConsumerIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataSetStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller**

*Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1198).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.273 src/main/activemq/wireformat/openwire/marshal/v5/ConsumerIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller**
*Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1206).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.274 src/main/activemq/wireformat/openwire/marshal/v1/ConsumerIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
```

7.275

src/main/activemq/wireformat/openwire/marshal/v2/ConsumerInfoMarshaller.h

File Reference

3391

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller**

*Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1226).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.275 src/main/activemq/wireformat/openwire/marshal/v2/ConsumerInfoMarshaller.h

File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller**

*Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1238).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.276 src/main/activemq/wireformat/openwire/marshal/v3/ConsumerInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller**

*Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1223).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.277 src/main/activemq/wireformat/openwire/marshal/v4/ConsumerInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
```


7.278

src/main/activemq/wireformat/openwire/marshal/v5/ConsumerInfoMarshaller.h

File Reference

3393

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataSetStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller**
*Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1230).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.278 src/main/activemq/wireformat/openwire/marshal/v5/ConsumerInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataSetStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller**
*Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1234).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.279 src/main/activemq/wireformat/openwire/marshal/v1/ControlCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller**

*Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1254).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.280 src/main/activemq/wireformat/openwire/marshal/v2/ControlCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
```

7.281 src/main/activemq/wireformat/openwire/marshal/v3/ControlCommandMarshaller.h File

Reference

3395

```
#include <decaf/io/DataInputStream.h>

#include <decaf/io/DataOutputStream.h>

#include <decaf/io/IOException.h>

#include <activemq/util/Config.h>

#include <activemq/commands/DataStructure.h>

#include <activemq/wireformat/openwire/OpenWireFormat.h>

#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller**

*Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1262).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.281 src/main/activemq/wireformat/openwire/marshal/v3/ControlCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>

#include <decaf/io/DataInputStream.h>

#include <decaf/io/DataOutputStream.h>

#include <decaf/io/IOException.h>

#include <activemq/util/Config.h>

#include <activemq/commands/DataStructure.h>

#include <activemq/wireformat/openwire/OpenWireFormat.h>

#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller**

*Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1246).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.282 src/main/activemq/wireformat/openwire/marshal/v4/ControlCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller**

*Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1250).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.283 src/main/activemq/wireformat/openwire/marshal/v5/ControlCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
```

7.284 src/main/activemq/wireformat/openwire/marsh shal/v1/DataArrayResponseMarshaller.h File

Reference

3397

```
#include <decaf/io/DataInputStream.h>

#include <decaf/io/DataOutputStream.h>

#include <decaf/io/IOException.h>

#include <activemq/util/Config.h>

#include <activemq/commands/DataStructure.h>

#include <activemq/wireformat/openwire/OpenWireFormat.h>

#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller**

*Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1258).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.284 src/main/activemq/wireformat/openwire/marsh shal/v1/DataArray File Reference

```
#include <activemq/wireformat/openwire/marsh/v1/ResponseMarshaller.h>

#include <decaf/io/DataInputStream.h>

#include <decaf/io/DataOutputStream.h>

#include <decaf/io/IOException.h>

#include <activemq/util/Config.h>

#include <activemq/commands/DataStructure.h>

#include <activemq/wireformat/openwire/OpenWireFormat.h>

#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller**

*Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1279).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.285 src/main/activemq/wireformat/openwire/marshal/v2/DataArray File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller**

*Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1287).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.286 src/main/activemq/wireformat/openwire/marshal/v3/DataArray File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/ResponseMarshaller.h>
```

7.287 src/main/activemq/wireformat/openwire/marsh- shal/v4/DataArrayResponseMarshaller.h File

Reference

3399

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller**

*Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1271).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.287 src/main/activemq/wireformat/openwire/marshal/v4/DataArray File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller**

*Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1275).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.288 src/main/activemq/wireformat/openwire/marshal/v5/DataArrayResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller**

*Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1283).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.289 src/main/activemq/wireformat/openwire/marshal/v1/DataResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/ResponseMarshaller.h>
```


7.290

src/main/activemq/wireformat/openwire/marshal/v2/DataResponseMarshaller.h
File Reference 3401

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller**
*Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1314).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.290 src/main/activemq/wireformat/openwire/marshal/v2/DataResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller**
*Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1326).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.291 src/main/activemq/wireformat/openwire/marshal/v3/DataResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller**
*Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1310).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.292 src/main/activemq/wireformat/openwire/marshal/v4/DataResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/ResponseMarshaller.h>
```

7.293

src/main/activemq/wireformat/openwire/marshal/v5/DataResponseMarshaller.h

File Reference

3403

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller**

*Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1322).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.293 src/main/activemq/wireformat/openwire/marshal/v5/DataResponseMarshaller.h

File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller**

*Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1318).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.294 src/main/activemq/wireformat/openwire/marshal/v1/DestinationInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller**
*Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1407).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.295 src/main/activemq/wireformat/openwire/marshal/v2/DestinationInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
```

7.296

src/main/activemq/wireformat/openwire/marshal/v3/DestinationInfoMarshaller.h
File Reference 3405

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataSetStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller**
*Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1395).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.296 src/main/activemq/wireformat/openwire/marshal/v3/DestinationInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataSetStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller**
*Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1399).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.297 src/main/activemq/wireformat/openwire/marshal/v4/Destination File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller**
*Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1403).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.298 src/main/activemq/wireformat/openwire/marshal/v5/Destination File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
```

7.299

src/main/activemq/wireformat/openwire/marshal/v1/DiscoveryEventMarshaller.h
File Reference 3407

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataSetStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller**
*Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1411).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.299 src/main/activemq/wireformat/openwire/marshal/v1/DiscoveryEventMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataSetStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller**
*Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1435).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.300 src/main/activemq/wireformat/openwire/marshal/v2/DiscoveryEventMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller**
*Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1420).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.301 src/main/activemq/wireformat/openwire/marshal/v3/DiscoveryEventMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
```


7.302

src/main/activemq/wireformat/openwire/marshal/v4/DiscoveryEventMarshaller.h
File Reference 3409

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataSetStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller**
*Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1424).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.302 src/main/activemq/wireformat/openwire/marshal/v4/DiscoveryEventMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataSetStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller**
*Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1427).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.303 src/main/activemq/wireformat/openwire/marshal/v5/DiscoveryEventMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller**
*Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1431).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.304 src/main/activemq/wireformat/openwire/marshal/v1/ExceptionMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/ResponseMarshaller.h>
```

7.305 src/main/activemq/wireformat/openwire/marsh shal/v2/ExceptionResponseMarshaller.h File

Reference

3411

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller**

*Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1491).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.305 src/main/activemq/wireformat/openwire/marsh shal/v2/ExceptionResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marsh/v2/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller**

*Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1487).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.306 src/main/activemq/wireformat/openwire/marshal/v3/ExceptionResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller**

*Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1495).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.307 src/main/activemq/wireformat/openwire/marshal/v4/ExceptionResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/ResponseMarshaller.h>
```

7.308 src/main/activemq/wireformat/openwire/marsh shal/v5/ExceptionResponseMarshaller.h File

Reference

3413

```
#include <decaf/io/DataInputStream.h>

#include <decaf/io/DataOutputStream.h>

#include <decaf/io/IOException.h>

#include <activemq/util/Config.h>

#include <activemq/commands/DataStructure.h>

#include <activemq/wireformat/openwire/OpenWireFormat.h>

#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller**

*Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1499).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.308 src/main/activemq/wireformat/openwire/marsh shal/v5/ExceptionResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marsh/v5/ResponseMarshaller.h>

#include <decaf/io/DataInputStream.h>

#include <decaf/io/DataOutputStream.h>

#include <decaf/io/IOException.h>

#include <activemq/util/Config.h>

#include <activemq/commands/DataStructure.h>

#include <activemq/wireformat/openwire/OpenWireFormat.h>

#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller**

*Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1503).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.309 src/main/activemq/wireformat/openwire/marshal/v1/FlushCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller**
*Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 1579).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.310 src/main/activemq/wireformat/openwire/marshal/v2/FlushCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
```

7.311

src/main/activemq/wireformat/openwire/marshal/v3/FlushCommandMarshaller.h
File Reference 3415

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataSetStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller**
*Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 1576).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.311 src/main/activemq/wireformat/openwire/marshal/v3/FlushCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataSetStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller**
*Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 1583).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.312 src/main/activemq/wireformat/openwire/marshal/v4/FlushCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller**
*Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 1587).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.313 src/main/activemq/wireformat/openwire/marshal/v5/FlushCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
```


7.314

src/main/activemq/wireformat/openwire/marshal/v1/IntegerResponseMarshaller.h
File Reference 3417

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataSetStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller**
*Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 1591).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.314 src/main/activemq/wireformat/openwire/marshal/v1/IntegerRes File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataSetStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller**
*Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 1673).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.315 src/main/activemq/wireformat/openwire/marshal/v2/IntegerRes File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller**
*Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 1669).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.316 src/main/activemq/wireformat/openwire/marshal/v3/IntegerRes File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/ResponseMarshaller.h>
```

7.317

src/main/activemq/wireformat/openwire/marshal/v4/IntegerResponseMarshaller.h
File Reference **3419**

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller**
*Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 1677).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.317 src/main/activemq/wireformat/openwire/marshal/v4/IntegerRes File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller**
*Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 1681).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.318 src/main/activemq/wireformat/openwire/marshal/v5/IntegerRes File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller**
*Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 1685).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.319 src/main/activemq/wireformat/openwire/marshal/v1/JournalQu File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
```

7.320 src/main/activemq/wireformat/openwire/marsh shal/v2/JournalQueueAckMarshaller.h File

Reference

3421

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller**

*Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 1733).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.320 src/main/activemq/wireformat/openwire/marsh shal/v2/JournalQueueAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marsh/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller**

*Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 1722).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.321 src/main/activemq/wireformat/openwire/marshal/v3/JournalQueueAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller**

*Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 1729).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.322 src/main/activemq/wireformat/openwire/marshal/v4/JournalQueueAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
```

7.323 src/main/activemq/wireformat/openwire/marshal/v5/JournalQueueAckMarshaller.h File

Reference

3423

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller**

*Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 1725).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.323 src/main/activemq/wireformat/openwire/marshal/v5/JournalQueueAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller**

*Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 1737).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.324 src/main/activemq/wireformat/openwire/marshal/v1/JournalTopicAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller**
*Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 1750).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.325 src/main/activemq/wireformat/openwire/marshal/v2/JournalTopicAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
```


7.326 src/main/activemq/wireformat/openwire/marshal/v3/JournalTopicAckMarshaller.h File

Reference

3425

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller**
*Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 1746).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.326 src/main/activemq/wireformat/openwire/marshal/v3/JournalTopicAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller**
*Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 1754).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.327 src/main/activemq/wireformat/openwire/marshal/v4/JournalTopicAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller**
*Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 1758).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.328 src/main/activemq/wireformat/openwire/marshal/v5/JournalTopicAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
```

7.329

src/main/activemq/wireformat/openwire/marshal/v1/JournalTraceMarshaller.h

File Reference

3427

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataSetStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller**
*Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 1762).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.329 src/main/activemq/wireformat/openwire/marshal/v1/JournalTraceMarshaller.h

File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataSetStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller**
*Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 1781).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.330 src/main/activemq/wireformat/openwire/marshal/v2/JournalTraceMarshallers.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller**
*Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 1769).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.331 src/main/activemq/wireformat/openwire/marshal/v3/JournalTraceMarshallers.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
```

7.332

src/main/activemq/wireformat/openwire/marshal/v4/JournalTraceMarshaller.h

File Reference

3429

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller**

*Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 1777).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.332 src/main/activemq/wireformat/openwire/marshal/v4/JournalTraceMarshaller.h

File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller**

*Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 1773).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.333 src/main/activemq/wireformat/openwire/marshal/v5/JournalTraceMarshallers.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller**
*Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 1785).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.334 src/main/activemq/wireformat/openwire/marshal/v1/JournalTraceMarshallers.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
```

7.335 src/main/activemq/wireformat/openwire/marsh shal/v2/JournalTransactionMarshaller.h File

Reference

3431

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller**

*Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 1804).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.335 src/main/activemq/wireformat/openwire/marsh shal/v2/JournalTransactionMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marsh/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller**

*Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 1793).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.336 src/main/activemq/wireformat/openwire/marshal/v3/JournalTransactionMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller**

*Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 1797).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.337 src/main/activemq/wireformat/openwire/marshal/v4/JournalTransactionMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
```


7.338 src/main/activemq/wireformat/openwire/marsh shal/v5/JournalTransactionMarshaller.h File

Reference

3433

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller**

*Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 1801).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.338 src/main/activemq/wireformat/openwire/marsh shal/v5/JournalTransactionMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marsh/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller**

*Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 1808).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.339 src/main/activemq/wireformat/openwire/marshal/v1/KeepAliveInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller**
*Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 1831).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.340 src/main/activemq/wireformat/openwire/marshal/v2/KeepAliveInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
```

7.341

src/main/activemq/wireformat/openwire/marshal/v3/KeepAliveInfoMarshaller.h

File Reference

3435

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller**
*Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 1815).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.341 src/main/activemq/wireformat/openwire/marshal/v3/KeepAliveInfoMarshaller.h

File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller**
*Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 1823).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.342 src/main/activemq/wireformat/openwire/marshal/v4/KeepAliveInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller**
*Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 1827).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.343 src/main/activemq/wireformat/openwire/marshal/v5/KeepAliveInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
```

7.344 src/main/activemq/wireformat/openwire/marsh shal/v1/LastPartialCommandMarshaller.h File

Reference

3437

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataSetStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller**
*Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 1819).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.344 src/main/activemq/wireformat/openwire/marsh shal/v1/LastPartial File Reference

```
#include <activemq/wireformat/openwire/marsh/v1/PartialCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataSetStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller**
*Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 1846).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.345 src/main/activemq/wireformat/openwire/marshal/v2/LastPartialCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/PartialCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller**

*Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 1842).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.346 src/main/activemq/wireformat/openwire/marshal/v3/LastPartialCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/PartialCommandMarshaller.h>
```

7.347 src/main/activemq/wireformat/openwire/marsh shal/v4/LastPartialCommandMarshaller.h File

Reference

3439

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller**

*Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 1850).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.347 src/main/activemq/wireformat/openwire/marsh shal/v4/LastPartial File Reference

```
#include <activemq/wireformat/openwire/marsh/v4/PartialCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller**

*Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 1854).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.348 src/main/activemq/wireformat/openwire/marshal/v5/LastPartialCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/PartialCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller**

*Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 1858).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.349 src/main/activemq/wireformat/openwire/marshal/v1/LocalTransactionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/TransactionIdMarshaller.h>
```


7.350 src/main/activemq/wireformat/openwire/marsh- shal/v2/LocalTransactionIdMarshaller.h File

Reference

3441

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataSetStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller**

*Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 1890).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.350 src/main/activemq/wireformat/openwire/marsh/v2/LocalTrans File Reference

```
#include <activemq/wireformat/openwire/marsh/v2/TransactionIdMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataSetStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller**

*Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 1878).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.351 src/main/activemq/wireformat/openwire/marshal/v3/LocalTransactionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/TransactionIdMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller**

*Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 1882).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.352 src/main/activemq/wireformat/openwire/marshal/v4/LocalTransactionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/TransactionIdMarshaller.h>
```

7.353 src/main/activemq/wireformat/openwire/marsh shal/v5/LocalTransactionIdMarshaller.h File

Reference

3443

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller**

*Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 1886).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.353 src/main/activemq/wireformat/openwire/marsh shal/v5/LocalTrans File Reference

```
#include <activemq/wireformat/openwire/marsh/v5/TransactionIdMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller**

*Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 1894).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.354 src/main/activemq/wireformat/openwire/marshal/v1/Marshaller File Reference

```
#include <activemq/wireformat/openwire/OpenWireFormat.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::MarshallerFactory**

Used to create marshallers for a specific version of the wire protocol.

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.355 src/main/activemq/wireformat/openwire/marshal/v2/Marshaller File Reference

```
#include <activemq/wireformat/openwire/OpenWireFormat.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::MarshallerFactory**

Used to create marshallers for a specific version of the wire protocol.

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.356 src/main/activemq/wireformat/openwire/marshal/v3/MarshallerFactory.h File Reference

```
#include <activemq/wireformat/openwire/OpenWireFormat.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::MarshallerFactory**

Used to create marshallers for a specific version of the wire protocol.

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.357 src/main/activemq/wireformat/openwire/marshal/v4/MarshallerFactory.h File Reference

```
#include <activemq/wireformat/openwire/OpenWireFormat.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::MarshallerFactory**

Used to create marshallers for a specific version of the wire protocol.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.358 src/main/activemq/wireformat/openwire/marshal/v5/Marshaller File Reference

```
#include <activemq/wireformat/openwire/OpenWireFormat.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::MarshallerFactory**
Used to create marshallers for a specific version of the wire protocol.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.359 src/main/activemq/wireformat/openwire/marshal/v1/MessageAc File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

7.360

src/main/activemq/wireformat/openwire/marshal/v2/MessageAckMarshaller.h File Reference 3447

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller**

*Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2076).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.360 src/main/activemq/wireformat/openwire/marshal/v2/MessageAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller**

*Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2060).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.361 src/main/activemq/wireformat/openwire/marshal/v3/MessageAck File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller**
*Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2068).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.362 src/main/activemq/wireformat/openwire/marshal/v4/MessageAck File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```


7.363

src/main/activemq/wireformat/openwire/marshal/v5/MessageAckMarshaller.h File Reference 3449
Data Structures

- class **activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller**

*Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2072).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.363 src/main/activemq/wireformat/openwire/marshal/v5/MessageAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller**

*Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 2064).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.364 src/main/activemq/wireformat/openwire/marshal/v1/MessageDispatchMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller**

*Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2107).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.365 src/main/activemq/wireformat/openwire/marshal/v2/MessageDispatchMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller**

*Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2095).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.366 src/main/activemq/wireformat/openwire/marshal/v3/MessageDi File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller**

*Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2103).*

Namespaces

- namespace **activemq**
*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.367 src/main/activemq/wireformat/openwire/marshal/v4/MessageDispatchMarshallers.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller**

*Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2099).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.368 src/main/activemq/wireformat/openwire/marshal/v5/MessageDispatchMarshallers.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller**

*Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 2111).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
 ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.369 src/main/activemq/wireformat/openwire/marshal/v1/MessageDi File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller**

*Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller**
 (p. 2132).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
 ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**

- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.370 src/main/activemq/wireformat/openwire/marshal/v2/MessageDi File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller**

*Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 2120).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.371 src/main/activemq/wireformat/openwire/marshal/v3/MessageDi File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
```

7.372 src/main/activemq/wireformat/openwire/marsh

Reference

3455

```
#include <activemq/commands/DataStructure.h>
```

```
#include <activemq/wireformat/openwire/OpenWireFormat.h>
```

```
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller**

Marshaling code for Open Wire Format for MessageDispatchNotificationMarshaller (p. 2128).

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.372 src/main/activemq/wireformat/openwire/marshal/v4/MessageDispatchNotificationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
```

```
#include <decaf/io/DataInputStream.h>
```

```
#include <decaf/io/DataOutputStream.h>
```

```
#include <decaf/io/IOException.h>
```

```
#include <activemq/util/Config.h>
```

```
#include <activemq/commands/DataStructure.h>
```

```
#include <activemq/wireformat/openwire/OpenWireFormat.h>
```

```
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller**

Marshaling code for Open Wire Format for MessageDispatchNotificationMarshaller (p. 2124).

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.373 src/main/activemq/wireformat/openwire/marshal/v5/MessageDispatchNotificationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller**

Marshaling code for Open Wire Format for MessageDispatchNotificationMarshaller (p. 2136).

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.374 src/main/activemq/wireformat/openwire/marshal/v1/MessageIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller**
*Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2161).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.375 src/main/activemq/wireformat/openwire/marshal/v2/MessageIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller**

*Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2146).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.376 src/main/activemq/wireformat/openwire/marshal/v3/MessageIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller**

*Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2154).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.377 src/main/activemq/wireformat/openwire/marshal/v4/MessageIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller**
*Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2150).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.378 src/main/activemq/wireformat/openwire/marshal/v5/MessageIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller**

*Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 2157).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.379 src/main/activemq/wireformat/openwire/marshal/v1/MessageM File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::MessageMarshaller**

*Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2183).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.380 src/main/activemq/wireformat/openwire/marshal/v2/MessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::MessageMarshaller**
*Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2166).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.381 src/main/activemq/wireformat/openwire/marshal/v3/MessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::MessageMarshaller**

*Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2175).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.382 src/main/activemq/wireformat/openwire/marshal/v4/MessageM File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::MessageMarshaller**

*Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2170).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.383 src/main/activemq/wireformat/openwire/marshal/v5/MessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::MessageMarshaller**
*Marshaling code for Open Wire Format for **MessageMarshaller** (p. 2179).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.384 src/main/activemq/wireformat/openwire/marshal/v1/MessagePublisher.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller**

*Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2215).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.385 src/main/activemq/wireformat/openwire/marshal/v2/MessagePu File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller**

*Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2207).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.386

src/main/activemq/wireformat/openwire/marshal/v3/MessagePullMarshaller.h

File Reference

7.386 src/main/activemq/wireformat/openwire/marshal/v3/MessagePu³⁴⁶⁵

File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller**

*Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2211).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.387 src/main/activemq/wireformat/openwire/marshal/v4/MessagePu

File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller**

*Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2223).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.388 src/main/activemq/wireformat/openwire/marshal/v5/MessagePu File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller**

*Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 2219).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.389 src/main/activemq/wireformat/openwire/marsh
shal/v1/NetworkBridgeFilterMarshaller.h File

Reference

7.389 src/main/activemq/wireformat/openwire/marsh³⁴⁶⁷shal/v1/NetworkBr

File Reference

```
#include <activemq/wireformat/openwire/marsh/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller**

*Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2261).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.390 src/main/activemq/wireformat/openwire/marsh/v2/NetworkBr

File Reference

```
#include <activemq/wireformat/openwire/marsh/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller**

*Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2250).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.391 src/main/activemq/wireformat/openwire/marshal/v3/NetworkBridgeFilterMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller**

*Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2253).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.392 src/main/activemq/wireformat/openwire/marsh
shal/v4/NetworkBridgeFilterMarshaller.h File

Reference

7.392 src/main/activemq/wireformat/openwire/marsh³⁴⁶⁹
shal/v4/NetworkBr

File Reference

```
#include <activemq/wireformat/openwire/marsh/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller**

*Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2265).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.393 src/main/activemq/wireformat/openwire/marsh/v5/NetworkBr

File Reference

```
#include <activemq/wireformat/openwire/marsh/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller**

*Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 2257).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.394 src/main/activemq/wireformat/openwire/marshal/v1/PartialCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller**

*Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2335).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.395 `src/main/activemq/wireformat/openwire/mar-`
`shal/v2/PartialCommandMarshaller.h` File

Reference

7.395 `src/main/activemq/wireformat/openwire/marsh`³⁴⁷¹ `al/v2/PartialCon` File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class `activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller`

*Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2322).*

Namespaces

- namespace `activemq`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `activemq::wireformat`
- namespace `activemq::wireformat::openwire`
- namespace `activemq::wireformat::openwire::marshal`
- namespace `activemq::wireformat::openwire::marshal::v2`

7.396 `src/main/activemq/wireformat/openwire/marshal/v3/PartialCon` File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller**

*Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2326).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.397 src/main/activemq/wireformat/openwire/marshal/v4/PartialCon File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller**

*Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2330).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.398 src/main/activemq/wireformat/openwire/marsh
shal/v5/PartialCommandMarshaller.h File

Reference

7.398 src/main/activemq/wireformat/openwire/marsh³⁴⁷³/v5/PartialCon File Reference

```
#include <activemq/wireformat/openwire/marshall/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller**

*Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2339).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.399 src/main/activemq/wireformat/openwire/marshall/v1/ProducerA File Reference

```
#include <activemq/wireformat/openwire/marshall/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller**

*Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 2439).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.400 src/main/activemq/wireformat/openwire/marshal/v2/ProducerAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller**

*Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 2424).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.401

src/main/activemq/wireformat/openwire/marshal/v3/ProducerAckMarshaller.h

File Reference

7.401 src/main/activemq/wireformat/openwire/marshal/v3/ProducerA³⁴⁷⁵

File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller**

*Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 2427).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.402 src/main/activemq/wireformat/openwire/marshal/v4/ProducerA

File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller**

*Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 2431).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.403 src/main/activemq/wireformat/openwire/marshal/v5/ProducerAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller**

*Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 2435).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.404

src/main/activemq/wireformat/openwire/marshal/v1/ProducerIdMarshaller.h File

Reference

7.404 src/main/activemq/wireformat/openwire/marshal/v1/ProducerIdMarshaller.h 3477

File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller**

*Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 2461).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.405 src/main/activemq/wireformat/openwire/marshal/v2/ProducerIdMarshaller.h File

File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller**

*Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 2450).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.406 src/main/activemq/wireformat/openwire/marshal/v3/ProducerIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller**

*Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 2454).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.407

src/main/activemq/wireformat/openwire/marshal/v4/ProducerIdMarshaller.h File

Reference

7.407 src/main/activemq/wireformat/openwire/marshal/v4/ProducerIdMarshaller.h 3479

File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller**

*Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 2465).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.408 src/main/activemq/wireformat/openwire/marshal/v5/ProducerIdMarshaller.h

File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller**

*Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 2457).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.409 src/main/activemq/wireformat/openwire/marshal/v1/ProducerInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller**

*Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 2490).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.410

src/main/activemq/wireformat/openwire/marshal/v2/ProducerInfoMarshaller.h

File Reference

7.410 src/main/activemq/wireformat/openwire/marshal/v2/ProducerInfoMarshaller.h³⁴⁸¹

File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller**

*Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 2474).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.411 src/main/activemq/wireformat/openwire/marshal/v3/ProducerInfoMarshaller.h

File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller**

*Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 2478).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.412 src/main/activemq/wireformat/openwire/marshal/v4/ProducerInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller**

*Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 2486).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.413

src/main/activemq/wireformat/openwire/marshal/v5/ProducerInfoMarshaller.h

File Reference

7.413 src/main/activemq/wireformat/openwire/marshal/v5/ProducerInfoMarshaller.h 3483

File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller**

*Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 2482).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.414 src/main/activemq/wireformat/openwire/marshal/v1/RemoveInfo

File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller**

*Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 2540).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.415 src/main/activemq/wireformat/openwire/marshal/v2/RemoveInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller**

*Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 2548).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.416

src/main/activemq/wireformat/openwire/marshal/v3/RemoveInfoMarshaller.h File
Reference

7.416 src/main/activemq/wireformat/openwire/marshal/v3/RemoveInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller**
*Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 2552).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.417 src/main/activemq/wireformat/openwire/marshal/v4/RemoveInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller**

*Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 2556).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.418 src/main/activemq/wireformat/openwire/marshal/v5/RemoveInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller**

*Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 2544).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.419 src/main/activemq/wireformat/openwire/marshal/v1/RemoveSubscriptionInfoMarshaller.h File

Reference

7.419 src/main/activemq/wireformat/openwire/marshal/v1/RemoveSubscriptionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller**

*Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 2572).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marsh**
- namespace **activemq::wireformat::openwire::marsh::v1**

7.420 src/main/activemq/wireformat/openwire/marshal/v2/RemoveSubscriptionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller**

*Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 2568).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.421 src/main/activemq/wireformat/openwire/marshal/v3/RemoveSubscriptionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller**

*Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 2576).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.422 src/main/activemq/wireformat/openwire/marsh- shal/v4/RemoveSubscriptionInfoMarshaller.h File

Reference

7.422 src/main/activemq/wireformat/openwire/marsh³⁴⁸⁹/v4/RemoveSu File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller**

*Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 2580).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.423 src/main/activemq/wireformat/openwire/marshal/v5/RemoveSu File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class `activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller`

*Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 2564).*

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::wireformat`
- namespace `activemq::wireformat::openwire`
- namespace `activemq::wireformat::openwire::marshal`
- namespace `activemq::wireformat::openwire::marshal::v5`

7.424 src/main/activemq/wireformat/openwire/marshal/v1/ReplayCom File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class `activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller`

*Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 2603).*

Namespaces

- namespace `activemq`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `activemq::wireformat`
- namespace `activemq::wireformat::openwire`
- namespace `activemq::wireformat::openwire::marshal`
- namespace `activemq::wireformat::openwire::marshal::v1`

7.425 src/main/activemq/wireformat/openwire/marsh
shal/v2/ReplayCommandMarshaller.h File

Reference

7.425 src/main/activemq/wireformat/openwire/marsh³⁴⁹¹ shal/v2/ReplayCon File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller**

*Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 2587).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.426 src/main/activemq/wireformat/openwire/marshal/v3/ReplayCon File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller**

*Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 2591).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.427 src/main/activemq/wireformat/openwire/marshal/v4/ReplayCom File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller**

*Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 2599).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.428 src/main/activemq/wireformat/openwire/marsh
shal/v5/ReplayCommandMarshaller.h File

Reference

7.428 src/main/activemq/wireformat/openwire/marsh³⁴⁹³/v5/ReplayCom File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller**

*Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 2595).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.429 src/main/activemq/wireformat/openwire/marshal/v1/ResponseM File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller**

*Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 2633).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.430 src/main/activemq/wireformat/openwire/marshal/v2/ResponseM File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller**

*Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 2620).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.431 src/main/activemq/wireformat/openwire/marshal/v3/ResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller**
*Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 2624).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.432 src/main/activemq/wireformat/openwire/marshal/v4/ResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ResponseMarshaller**

*Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 2637).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.433 src/main/activemq/wireformat/openwire/marshal/v5/ResponseM File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ResponseMarshaller**

*Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 2628).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.434 src/main/activemq/wireformat/openwire/marshal/v1/SessionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller**
*Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 2685).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.435 src/main/activemq/wireformat/openwire/marshal/v2/SessionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller**

*Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 2689).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.436 src/main/activemq/wireformat/openwire/marshal/v3/SessionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller**

*Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 2697).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.437 src/main/activemq/wireformat/openwire/marshal/v4/SessionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller**
*Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 2693).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.438 src/main/activemq/wireformat/openwire/marshal/v5/SessionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller**

*Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 2681).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.439 src/main/activemq/wireformat/openwire/marshal/v1/SessionInfo File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller**

*Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 2709).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.440

src/main/activemq/wireformat/openwire/marshal/v2/SessionInfoMarshaller.h File

Reference

7.440 src/main/activemq/wireformat/openwire/marshal/v2/SessionInfo

File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller**

*Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 2705).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.441 src/main/activemq/wireformat/openwire/marshal/v3/SessionInfo

File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller**

*Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 2721).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.442 src/main/activemq/wireformat/openwire/marshal/v4/SessionInfo File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller**

*Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 2713).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.443

src/main/activemq/wireformat/openwire/marshal/v5/SessionInfoMarshaller.h File

Reference

7.443 src/main/activemq/wireformat/openwire/marshal/v5/SessionInfo

File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller**

*Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 2717).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.444 src/main/activemq/wireformat/openwire/marshal/v1/ShutdownI

File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller**

*Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 2760).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.445 src/main/activemq/wireformat/openwire/marshal/v2/ShutdownInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller**

*Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 2771).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.446

src/main/activemq/wireformat/openwire/marshal/v3/ShutdownInfoMarshaller.h

File Reference

7.446 src/main/activemq/wireformat/openwire/marshal/v3/ShutdownInfoMarshaller.h 3505

File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller**

*Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 2767).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.447 src/main/activemq/wireformat/openwire/marshal/v4/ShutdownInfoMarshaller.h

File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller**
*Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 2763).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.448 src/main/activemq/wireformat/openwire/marshal/v5/ShutdownInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller**
*Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 2775).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.449

src/main/activemq/wireformat/openwire/marshal/v1/SubscriptionInfoMarshaller.h

File Reference

7.449 ³⁵⁰⁷ src/main/activemq/wireformat/openwire/marshal/v1/SubscriptionInfoMarshaller.h

File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller**

*Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 2911).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.450 src/main/activemq/wireformat/openwire/marshal/v2/SubscriptionInfoMarshaller.h

File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller**
*Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 2926).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.451 src/main/activemq/wireformat/openwire/marshal/v3/SubscriptionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utills/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller**
*Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 2915).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.452

src/main/activemq/wireformat/openwire/marshal/v4/SubscriptionInfoMarshaller.h

File Reference

7.452 src/main/activemq/wireformat/openwire/marshal/v4/SubscriptionInfoMarshaller.h 3509

File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller**

*Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 2923).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.453 src/main/activemq/wireformat/openwire/marshal/v5/SubscriptionInfoMarshaller.h

File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller**
*Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 2919).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.454 src/main/activemq/wireformat/openwire/marshal/v1/TransactionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller**
*Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3022).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.455

src/main/activemq/wireformat/openwire/marshal/v2/TransactionIdMarshaller.h

File Reference

7.455 src/main/activemq/wireformat/openwire/marshal/v2/TransactionIdMarshaller.h³⁵¹¹

File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller**

*Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3018).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.456 src/main/activemq/wireformat/openwire/marshal/v3/TransactionIdMarshaller.h

File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller**

*Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3033).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.457 src/main/activemq/wireformat/openwire/marshal/v4/TransactionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller**

*Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3026).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.458

src/main/activemq/wireformat/openwire/marshal/v5/TransactionIdMarshaller.h

File Reference

7.458 src/main/activemq/wireformat/openwire/marshal/v5/TransactionIdMarshaller.h 3513

File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller**

*Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 3029).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.459 src/main/activemq/wireformat/openwire/marshal/v1/TransactionIdMarshaller.h

File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller**
*Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3048).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.460 src/main/activemq/wireformat/openwire/marshal/v2/TransactionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller**
*Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3056).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.461

src/main/activemq/wireformat/openwire/marshal/v3/TransactionInfoMarshaller.h

File Reference

7.461 src/main/activemq/wireformat/openwire/marshal/v3/TransactionInfoMarshaller.h

File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller**

*Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3044).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.462 src/main/activemq/wireformat/openwire/marshal/v4/TransactionInfoMarshaller.h

File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller**
*Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3052).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.463 src/main/activemq/wireformat/openwire/marshal/v5/TransactionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller**
*Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 3041).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.464

src/main/activemq/wireformat/openwire/marshal/v1/WireFormatInfoMarshaller.h

File Reference

7.464 src/main/activemq/wireformat/openwire/marshal/v1/WireFormatInfoMarshaller.h

File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller**

*Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3170).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.465 src/main/activemq/wireformat/openwire/marshal/v2/WireFormatInfoMarshaller.h

File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller**
*Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3163).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.466 src/main/activemq/wireformat/openwire/marshal/v3/WireFormatInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller**
*Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3178).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.467

src/main/activemq/wireformat/openwire/marshal/v4/WireFormatInfoMarshaller.h

File Reference

7.467 src/main/activemq/wireformat/openwire/marshal/v4/WireFormatInfoMarshaller.h 3519

File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller**

*Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3174).*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.468 src/main/activemq/wireformat/openwire/marshal/v5/WireFormatInfoMarshaller.h

File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller**
*Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 3166).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.469 src/main/activemq/wireformat/openwire/marshal/v1/XATransactionIdMarshaller File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/TransactionIdMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller**
*Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3209).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.470 src/main/activemq/wireformat/openwire/marsh
shal/v2/XATransactionIdMarshaller.h File

Reference

7.470 src/main/activemq/wireformat/openwire/marsh³⁵²¹/v2/XATransac File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/TransactionIdMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller**

*Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3197).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.471 src/main/activemq/wireformat/openwire/marshal/v3/XATransac File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/TransactionIdMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller**

*Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3201).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.472 src/main/activemq/wireformat/openwire/marshal/v4/XATransactionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v4/TransactionIdMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/Utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller**

*Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3205).*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v4**

7.473 src/main/activemq/wireformat/openwire/marsh-
shal/v5/XATransactionIdMarshaller.h File

Reference

7.473 ³⁵²³src/main/activemq/wireformat/openwire/marshal/v5/XATransac
File Reference

```
#include <activemq/wireformat/openwire/marshal/v5/TransactionIdMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller**

*Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 3213).*

Namespaces

- namespace **activemq**

*Licensed to the Apache Software Foundation (ASF) under one or more contributor license agree-
ments.*

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v5**

7.474 src/main/activemq/wireformat/openwire/OpenWireFormat.h
File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/WireFormatInfo.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/WireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/Properties.h>
#include <decaf/util/concurrent/atomic/AtomicBoolean.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
```

```
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <memory>
```

Data Structures

- class **activemq::wireformat::openwire::OpenWireFormat**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**

7.475 src/main/activemq/wireformat/openwire/OpenWireFormatFactory File Reference

```
#include <activemq/util/Config.h>
#include <activemq/wireformat/WireFormatFactory.h>
#include <activemq/commands/WireFormatInfo.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/Properties.h>
```

Data Structures

- class **activemq::wireformat::openwire::OpenWireFormatFactory**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**

7.476 src/main/activemq/wireformat/openwire/OpenWireFormatNegot File Reference

```
#include <activemq/util/Config.h>
```

```
#include <activemq/transport/TransportFilter.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/WireFormatNegotiator.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/concurrent/CountDownLatch.h>
#include <decaf/util/concurrent/Concurrent.h>
#include <decaf/util/concurrent/atomic/AtomicBoolean.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::wireformat::openwire::OpenWireFormatNegotiator**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**

7.477 src/main/activemq/wireformat/openwire/OpenWireResponseBuilder.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/mock/ResponseBuilder.h>
#include <decaf/util/StlQueue.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::wireformat::openwire::OpenWireResponseBuilder**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**

7.478 src/main/activemq/wireformat/openwire/utils/BooleanStream.h File Reference

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::wireformat::openwire::utils::BooleanStream**

Manages the writing and reading of boolean data streams to and from a data source such as a `DataInputStream` or `DataOutputStream`.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::utils**

7.479 src/main/activemq/wireformat/openwire/utils/HexTable.h File Reference

```
#include <vector>
#include <string>
#include <activemq/util/Config.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
```

Data Structures

- class **activemq::wireformat::openwire::utils::HexTable**

*The **HexTable** (p. 1609) class maps hexadecimal strings to the value of an index into the table, i.e.*

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::utls**

7.480 src/main/activemq/wireformat/openwire/utls/MessagePropertyI File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/Message.h>
#include <activemq/util/PrimitiveMap.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
```

Data Structures

- class **activemq::wireformat::openwire::utls::MessagePropertyInterceptor**
Used the base ActiveMQMessage class to intercept calls to get and set properties in order to capture the calls that use the reserved JMS properties and get and set them in the OpenWire Message properties.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::utls**

7.481 src/main/activemq/wireformat/openwire/utls/OpenwireStringSu File Reference

```
#include <string>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::wireformat::openwire::utls::OpenwireStringSupport**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::utils**

7.482 src/main/activemq/wireformat/stomp/StompCommandConstants.h File Reference

```
#include <cms/Destination.h>
#include <activemq/util/Config.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <string>
#include <map>
```

Data Structures

- class **activemq::wireformat::stomp::StompCommandConstants**

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::stomp**

7.483 src/main/activemq/wireformat/stomp/StompFrame.h File Reference

```
#include <string>
#include <string.h>
#include <map>
#include <decaf/util/Properties.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/DataInputStream.h>
#include <activemq/util/Config.h>
```


Data Structures

- class **activemq::wireformat::stomp::StompFrame**
A Stomp-level message frame that encloses all messages to and from the broker.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::stomp**

7.484 src/main/activemq/wireformat/stomp/StompHelper.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/util/LongSequenceGenerator.h>
#include <activemq/wireformat/stomp/StompFrame.h>
#include <activemq/commands/Message.h>
#include <activemq/commands/MessageId.h>
#include <activemq/commands/ProducerId.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/TransactionId.h>
#include <activemq/commands/ActiveMQDestination.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::wireformat::stomp::StompHelper**
Utility Methods used when marshaling to and from StompFrame's.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::stomp**

7.485 src/main/activemq/wireformat/stomp/StompWireFormat.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/wireformat/WireFormat.h>
#include <activemq/wireformat/stomp/StompFrame.h>
#include <activemq/wireformat/stomp/StompHelper.h>
#include <decaf/util/concurrent/atomic/AtomicBoolean.h>
#include <decaf/io/IOException.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::wireformat::stomp::StompWireFormat**

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::stomp**

7.486 src/main/activemq/wireformat/stomp/StompWireFormatFactory.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/wireformat/WireFormatFactory.h>
#include <activemq/wireformat/stomp/StompWireFormat.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::wireformat::stomp::StompWireFormatFactory**
Factory used to create the Stomp Wire Format instance.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::stomp**

7.487 src/main/activemq/wireformat/WireFormat.h File Reference

```
#include <activemq/wireformat/WireFormatNegotiator.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <decaf/lang/Pointer.h>
#include <activemq/util/Config.h>
#include <activemq/commands/Command.h>
#include <activemq/transport/Transport.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
```

Data Structures

- class **activemq::wireformat::WireFormat**
Provides a mechanism to marshal commands into and out of packets or into and out of streams, Channels and Datagrams.

Namespaces

- namespace **activemq**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**

7.488 src/main/activemq/wireformat/WireFormatFactory.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/wireformat/WireFormat.h>
#include <decaf/util/Properties.h>
#include <decaf/lang/Pointer.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
```

Data Structures

- class **activemq::wireformat::WireFormatFactory**

*The **WireFormatFactory** (p. 3151) is the interface that all **WireFormatFactory** (p. 3151) classes must extend.*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**

7.489 src/main/activemq/wireformat/WireFormatNegotiator.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/TransportFilter.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::wireformat::WireFormatNegotiator**

*Defines a **WireFormatNegotiator** (p. 3182) which allows a **WireFormat** (p. 3148) to.*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**

7.490 src/main/activemq/wireformat/WireFormatRegistry.h File Reference

```
#include <activemq/util/Config.h>
#include <string>
#include <vector>
#include <activemq/wireformat/WireFormatFactory.h>
#include <decaf/util/StlMap.h>
```

```
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
```

Data Structures

- class **activemq::wireformat::WireFormatRegistry**

*Registry of all **WireFormat** (p. 3148) Factories that are available to the client at runtime.*

Namespaces

- namespace **activemq**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**

7.491 src/main/cms/BytesMessage.h File Reference

```
#include <cms/Config.h>
#include <cms/Message.h>
#include <cms/CMSException.h>
#include <cms/MessageNotReadableException.h>
#include <cms/MessageNotWritableException.h>
#include <cms/MessageEOFException.h>
#include <cms/MessageFormatException.h>
```

Data Structures

- class **cms::BytesMessage**

*A **BytesMessage** (p. 874) object is used to send a message containing a stream of unsigned bytes.*

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.492 src/main/cms/Closeable.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

Data Structures

- class **cms::Closeable**
Interface for a class that implements the close method.

Namespaces

- namespace **cms**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.493 src/main/decaf/io/Closeable.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/IOException.h>
```

Data Structures

- class **decaf::io::Closeable**
Interface for a class that implements the close method.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::io**

7.494 src/main/cms/CMSException.h File Reference

```
#include <string>
#include <vector>
#include <iostream>
#include <exception>
#include <cms/Config.h>
```

Data Structures

- class **cms::CMSException**

CMS API Exception that is the base for all exceptions thrown from CMS classes.

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.495 src/main/cms/CMSProperties.h File Reference

```
#include <cms/Config.h>
#include <map>
#include <string>
#include <vector>
```

Data Structures

- class **cms::CMSProperties**

Interface for a Java-like properties object.

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.496 src/main/cms/CMSSecurityException.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

Data Structures

- class **cms::CMSSecurityException**

This exception must be thrown when a provider rejects a user name/password submitted by a client.

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.497 src/main/cms/Connection.h File Reference

```
#include <cms/Config.h>
#include <cms/Startable.h>
#include <cms/Stopable.h>
#include <cms/Closeable.h>
#include <cms/Session.h>
#include <cms/ConnectionMetaData.h>
```

Data Structures

- class **cms::Connection**

The client's connection to its provider.

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.498 src/main/cms/ConnectionFactory.h File Reference

```
#include <cms/Config.h>
#include <cms/Connection.h>
#include <cms/CMSException.h>
#include <string>
```

Data Structures

- class **cms::ConnectionFactory**

*Defines the interface for a factory that creates connection objects, the **Connection** (p. 1052) objects returned implement the **CMS Connection** (p. 1052) interface and hide the CMS Provider specific implementation details behind that interface.*

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.499 src/main/cms/ConnectionMetaData.h File Reference

```
#include <cms/Config.h>
```

```
#include <cms/CMSException.h>
```

Data Structures

- class **cms::ConnectionMetaData**

*A **ConnectionMetaData** (p. 1154) object provides information describing the **Connection** (p. 1052) object.*

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.500 src/main/cms/DeliveryMode.h File Reference

```
#include <cms/Config.h>
```

Data Structures

- class **cms::DeliveryMode**

This is an Abstract class whose purpose is to provide a container for the delivery mode enumeration for CMS messages.

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.501 src/main/cms/Destination.h File Reference

```
#include <cms/CMSProperties.h>
#include <cms/Config.h>
#include <string>
```

Data Structures

- class **cms::Destination**

*A **Destination** (p. 1387) object encapsulates a provider-specific address.*

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.502 src/main/cms/ExceptionListener.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

Data Structures

- class **cms::ExceptionListener**

*If a CMS provider detects a serious problem, it notifies the client application through an **ExceptionListener** (p. 1483) that is registered with the **Connection** (p. 1052).*

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.503 src/main/cms/IllegalStateException.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

Data Structures

- class **cms::IllegalStateException**

This exception is thrown when a method is invoked at an illegal or inappropriate time or if the provider is not in an appropriate state for the requested operation.

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.504 src/main/decaf/lang/exceptions/IllegalStateException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::lang::exceptions::IllegalStateException**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

7.505 src/main/cms/InvalidClientIdException.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

Data Structures

- class **cms::InvalidClientIdException**

This exception must be thrown when a client attempts to set a connection's client ID to a value that is rejected by a provider.

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.506 src/main/cms/InvalidDestinationException.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

Data Structures

- class **cms::InvalidDestinationException**

This exception must be thrown when a destination either is not understood by a provider or is no longer valid.

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.507 src/main/cms/InvalidSelectorException.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

Data Structures

- class **cms::InvalidSelectorException**

This exception must be thrown when a CMS client attempts to give a provider a message selector with invalid syntax.

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.508 src/main/cms/MapMessage.h File Reference

```
#include <cms/Config.h>
#include <cms/Message.h>
#include <cms/CMSException.h>
#include <cms/MessageFormatException.h>
#include <cms/MessageNotWriteableException.h>
```

Data Structures

- class **cms::MapMessage**

*A **MapMessage** (p. 1982) object is used to send a set of name-value pairs.*

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.509 src/main/cms/MessageConsumer.h File Reference

```
#include <cms/Config.h>
#include <cms/MessageListener.h>
#include <cms/Message.h>
#include <cms/Closeable.h>
```

Data Structures

- class **cms::MessageConsumer**

*A client uses a **MessageConsumer** (p. 2080) to received messages from a destination.*

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.510 src/main/cms/MessageEOFException.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSEException.h>
```

Data Structures

- class **cms::MessageEOFException**

*This exception must be thrown when an unexpected end of stream has been reached when a **StreamMessage** (p. 2892) or **BytesMessage** (p. 874) is being read.*

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.511 src/main/cms/MessageFormatException.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSEException.h>
```

Data Structures

- class **cms::MessageFormatException**

This exception must be thrown when a CMS client attempts to use a data type not supported by a message or attempts to read data in a message as the wrong type.

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.512 src/main/cms/MessageListener.h File Reference

```
#include <cms/Config.h>
```

Data Structures

- class **cms::MessageListener**

A MessageListener (p. 2165) object is used to receive asynchronously delivered messages.

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.513 src/main/cms/MessageNotReadableException.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

Data Structures

- class **cms::MessageNotReadableException**

This exception must be thrown when a CMS client attempts to read a write-only message.

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.514 src/main/cms/MessageNotWriteableException.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

Data Structures

- class **cms::MessageNotWriteableException**

This exception must be thrown when a CMS client attempts to write to a read-only message.

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.515 src/main/cms/MessageProducer.h File Reference

```
#include <cms/Config.h>
#include <cms/Message.h>
#include <cms/Destination.h>
#include <cms/Closeable.h>
#include <cms/CMSException.h>
#include <cms/InvalidDestinationException.h>
#include <cms/MessageFormatException.h>
#include <cms/UnsupportedOperationException.h>
#include <cms/DeliveryMode.h>
```

Data Structures

- class **cms::MessageProducer**

*A client uses a **MessageProducer** (p. 2189) object to send messages to a **Destination** (p. 1387).*

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.516 src/main/cms/ObjectMessage.h File Reference

```
#include <cms/Config.h>
#include <cms/Message.h>
```

Data Structures

- class **cms::ObjectMessage**

Place holder for interaction with JMS systems that support Java, the C++ client is not responsible for deserializing the contained Object.

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.517 src/main/cms/Queue.h File Reference

```
#include <cms/Config.h>
#include <cms/Destination.h>
#include <cms/CMSException.h>
```

Data Structures

- class **cms::Queue**

An interface encapsulating a provider-specific queue name.

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.518 src/main/decaf/util/Queue.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/AbstractCollection.h>
#include <decaf/lang/Exception.h>
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
```

Data Structures

- class **decaf::util::Queue< E >**

A kind of collection provides advanced operations than other basic collections, such as insertion, extraction, and inspection.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**

7.519 src/main/cms/QueueBrowser.h File Reference

```
#include <vector>
#include <string>
#include <cms/Config.h>
#include <cms/Closeable.h>
#include <cms/Queue.h>
#include <cms/Message.h>
#include <cms/CMSException.h>
```

Data Structures

- class **cms::QueueBrowser**

*This class implements in interface for browsing the messages in a **Queue** (p. 2510) without removing them.*

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.520 src/main/cms/Session.h File Reference

```
#include <cms/Config.h>
#include <cms/Closeable.h>
#include <cms/Message.h>
#include <cms/TextMessage.h>
#include <cms/BytesMessage.h>
#include <cms/MapMessage.h>
#include <cms/StreamMessage.h>
#include <cms/MessageProducer.h>
#include <cms/MessageConsumer.h>
```

```
#include <cms/Topic.h>
#include <cms/Queue.h>
#include <cms/QueueBrowser.h>
#include <cms/TemporaryTopic.h>
#include <cms/TemporaryQueue.h>
#include <cms/CMSException.h>
```

Data Structures

- class **cms::Session**

*A **Session** (p. 2663) object is a single-threaded context for producing and consuming messages.*

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.521 src/main/cms/Startable.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

Data Structures

- class **cms::Startable**

Interface for a class that implements the start method.

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.522 src/main/cms/Stopable.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

Data Structures

- class **cms::Stoppable**

Interface for a class that implements the stop method.

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.523 src/main/cms/StreamMessage.h File Reference

```
#include <cms/Config.h>
#include <cms/Message.h>
#include <cms/CMSException.h>
#include <cms/MessageNotReadableException.h>
#include <cms/MessageNotWriteableException.h>
#include <cms/MessageFormatException.h>
#include <cms/MessageEOFException.h>
```

Data Structures

- class **cms::StreamMessage**

*Interface for a **StreamMessage** (p. 2892).*

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.524 src/main/cms/TemporaryQueue.h File Reference

```
#include <cms/Config.h>
#include <cms/Destination.h>
#include <cms/CMSException.h>
```

Data Structures

- class **cms::TemporaryQueue**

*Defines a Temporary **Queue** (p. 2510) based **Destination** (p. 1387).*

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.525 src/main/cms/TemporaryTopic.h File Reference

```
#include <cms/Config.h>
#include <cms/Destination.h>
#include <cms/CMSException.h>
```

Data Structures

- class **cms::TemporaryTopic**

*Defines a Temporary **Topic** (p. 3014) based **Destination** (p. 1387).*

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.526 src/main/cms/TextMessage.h File Reference

```
#include <cms/Config.h>
#include <cms/Message.h>
#include <cms/CMSException.h>
#include <cms/MessageNotWriteableException.h>
```

Data Structures

- class **cms::TextMessage**

Interface for a text message.

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.527 src/main/cms/Topic.h File Reference

```
#include <cms/Config.h>
#include <cms/Destination.h>
#include <cms/CMSException.h>
```

Data Structures

- class **cms::Topic**

An interface encapsulating a provider-specific topic name.

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.528 src/main/cms/UnsupportedOperationException.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

Data Structures

- class **cms::UnsupportedOperationException**

This exception must be thrown when a CMS client attempts use a CMS method that is not implemented or not supported by the CMS Provider in use.

Namespaces

- namespace **cms**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.529 src/main/decaf/lang/exceptions/UnsupportedOperationException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::lang::exceptions::UnsupportedOperationException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

7.530 src/main/decaf/internal/AprPool.h File Reference

```
#include <decaf/util/Config.h>  
#include <apr_pools.h>
```

Data Structures

- class **decaf::internal::AprPool**
Wraps an APR pool object so that classes in decaf can create a static member for use in static methods where apr function calls that need a pool are made.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**

7.531 src/main/decaf/internal/DecafRuntime.h File Reference

```
#include <decaf/util/Config.h>  
#include <decaf/lang/Runtime.h>  
#include <memory>  
#include <apr_pools.h>
```

Data Structures

- class **decaf::internal::DecafRuntime**
Handles APR initialization and termination.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**

7.532 **src/main/decaf/internal/io/StandardErrorOutputStream.h** File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/OutputStream.h>
#include <decaf/util/concurrent/Mutex.h>
```

Data Structures

- class **decaf::internal::io::StandardErrorOutputStream**
Wrapper Around the Standard error Output facility on the current platform.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::io**

7.533 **src/main/decaf/internal/io/StandardInputStream.h** File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/InputStream.h>
#include <decaf/util/concurrent/Mutex.h>
```


Data Structures

- class **decaf::internal::io::StandardInputStream**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::internal**
- namespace **decaf::internal::io**

7.534 src/main/decaf/internal/io/StandardOutputStream.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/OutputStream.h>
#include <decaf/util/concurrent/Mutex.h>
```

Data Structures

- class **decaf::internal::io::StandardOutputStream**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::internal**
- namespace **decaf::internal::io**

7.535 src/main/decaf/internal/net/URLEncoderDecoder.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/URISyntaxException.h>
#include <string>
```

Data Structures

- class **decaf::internal::net::URLEncoderDecoder**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::internal**
- namespace **decaf::internal::net**

7.536 src/main/decaf/internal/net/URIHelper.h File Reference

```
#include <string>
#include <decaf/util/Config.h>
#include <decaf/net/URISyntaxException.h>
#include <decaf/internal/net/URIType.h>
```

Data Structures

- class **decaf::internal::net::URIHelper**

Helper class used by the URI classes in encoding and decoding of URI's.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::internal**
- namespace **decaf::internal::net**

7.537 src/main/decaf/internal/net/URIType.h File Reference

```
#include <string>
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::internal::net::URIType**

Basic type object that holds data that composes a given URI.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::internal**
- namespace **decaf::internal::net**

7.538 src/main/decaf/internal/nio/BufferFactory.h File Reference

```
#include <decaf/nio/ByteBuffer.h>
#include <decaf/nio/CharBuffer.h>
#include <decaf/nio/DoubleBuffer.h>
#include <decaf/nio/FloatBuffer.h>
#include <decaf/nio/LongBuffer.h>
#include <decaf/nio/IntBuffer.h>
#include <decaf/nio/ShortBuffer.h>
```

Data Structures

- class **decaf::internal::nio::BufferFactory**

*Factory class used by static methods in the **decaf::nio** (p. 109) package to create the various default version of the NIO interfaces.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::internal**
- namespace **decaf::internal::nio**

7.539 src/main/decaf/internal/nio/ByteArrayBuffer.h File Reference

```
#include <decaf/nio/ByteBuffer.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
```

```
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
#include <decaf/internal/nio/ByteArrayPerspective.h>
#include <decaf/nio/CharBuffer.h>
#include <decaf/nio/DoubleBuffer.h>
#include <decaf/nio/FloatBuffer.h>
#include <decaf/nio/ShortBuffer.h>
#include <decaf/nio/IntBuffer.h>
#include <decaf/nio/LongBuffer.h>
```

Data Structures

- class **decaf::internal::nio::ByteBuffer**

This class defines six categories of operations upon byte buffers:

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::internal**
- namespace **decaf::internal::nio**

7.540 src/main/decaf/internal/nio/ByteArrayPerspective.h File Reference

```
#include <decaf/internal/util/ByteArrayAdapter.h>
```

Data Structures

- class **decaf::internal::nio::ByteArrayPerspective**

This class extends `ByteArray` to create a reference counted byte array that can be held and used by several different `ByteBuffers` and allow them to know on destruction whose job it is to delete the perspective.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::internal**

- namespace **decaf::internal::nio**

7.541 src/main/decaf/internal/nio/CharArrayBuffer.h File Reference

```
#include <decaf/nio/CharBuffer.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
#include <decaf/internal/nio/ByteArrayPerspective.h>
```

Data Structures

- class **decaf::internal::nio::CharArrayBuffer**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::internal**
- namespace **decaf::internal::nio**

7.542 src/main/decaf/internal/nio/DoubleArrayBuffer.h File Reference

```
#include <decaf/nio/DoubleBuffer.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
#include <decaf/internal/nio/ByteArrayPerspective.h>
```

Data Structures

- class **decaf::internal::nio::DoubleArrayBuffer**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::internal**
- namespace **decaf::internal::nio**

7.543 src/main/decaf/internal/nio/FloatArrayBuffer.h File Reference

```
#include <decaf/nio/FloatBuffer.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
#include <decaf/internal/nio/ByteArrayPerspective.h>
```

Data Structures

- class **decaf::internal::nio::FloatArrayBuffer**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::internal**
- namespace **decaf::internal::nio**

7.544 src/main/decaf/internal/nio/IntArrayBuffer.h File Reference

```
#include <decaf/nio/IntBuffer.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
#include <decaf/internal/nio/ByteArrayPerspective.h>
```

Data Structures

- class `decaf::internal::nio::IntArrayBuffer`

Namespaces

- namespace `decaf`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `decaf::internal`
- namespace `decaf::internal::nio`

7.545 src/main/decaf/internal/nio/LongArrayBuffer.h File Reference

```
#include <decaf/nio/LongBuffer.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
#include <decaf/internal/nio/ByteArrayPerspective.h>
```

Data Structures

- class `decaf::internal::nio::LongArrayBuffer`

Namespaces

- namespace `decaf`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `decaf::internal`
- namespace `decaf::internal::nio`

7.546 src/main/decaf/internal/nio/ShortArrayBuffer.h File Reference

```
#include <decaf/nio/ShortBuffer.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
```

```
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
#include <decaf/internal/nio/ByteArrayPerspective.h>
```

Data Structures

- class **decaf::internal::nio::ShortArrayBuffer**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::internal**
- namespace **decaf::internal::nio**

7.547 src/main/decaf/internal/util/ByteArrayAdapter.h File Reference

```
#include <decaf/lang/exceptions/InvalidStateException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
```

Data Structures

- class **decaf::internal::util::ByteArrayAdapter**

This class adapts primitive type arrays to a base byte array so that the classes can inter-operate on the same base byte array without copying data.

- union **decaf::internal::util::ByteArrayAdapter::Array**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::internal**
- namespace **decaf::internal::util**

7.548 src/main/decaf/internal/util/concurrent/ConditionImpl.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::internal::util::concurrent::ConditionImpl**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::internal**
- namespace **decaf::internal::util**
- namespace **decaf::internal::util::concurrent**

7.549 src/main/decaf/internal/util/concurrent/MutexImpl.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::internal::util::concurrent::MutexImpl**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::internal**
- namespace **decaf::internal::util**
- namespace **decaf::internal::util::concurrent**

7.550 src/main/decaf/internal/util/concurrent/SynchronizableImpl.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/concurrent/Mutex.h>
```

Data Structures

- class **decaf::internal::util::concurrent::SynchronizableImpl**

A convenience class used by some Decaf classes to implement the Synchronizable interface when there is no issues related to multiple inheritance.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::util**
- namespace **decaf::internal::util::concurrent**

7.551 src/main/decaf/internal/util/concurrent/Transferer.h File Reference

```
#include <decaf/lang/exceptions/InterruptedException.h>
#include <decaf/util/concurrent/TimeoutException.h>
```

Data Structures

- class **decaf::internal::util::concurrent::Transferer< E >**

Shared internal API for dual stacks and queues.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::util**
- namespace **decaf::internal::util::concurrent**

7.552 src/main/decaf/internal/util/concurrent/TransferQueue.h File Reference

```
#include <decaf/internal/util/concurrent/Transferer.h>
#include <decaf/util/concurrent/locks/LockSupport.h>
#include <decaf/util/concurrent/atomic/AtomicReference.h>
#include <decaf/util/concurrent/TimeoutException.h>
#include <decaf/lang/exceptions/InterruptedException.h>
#include <decaf/lang/Thread.h>
```

Data Structures

- class **decaf::internal::util::concurrent::TransferQueue< E >**
This extends Scherer-Scott dual queue algorithm, differing, among other ways, by using modes within nodes rather than marked pointers.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::util**
- namespace **decaf::internal::util::concurrent**

7.553 src/main/decaf/internal/util/concurrent/TransferStack.h File Reference

```
#include <decaf/internal/util/concurrent/Transferer.h>
#include <decaf/util/concurrent/TimeoutException.h>
#include <decaf/lang/exceptions/InterruptedException.h>
```

Data Structures

- class **decaf::internal::util::concurrent::TransferStack< E >**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::internal**
- namespace **decaf::internal::util**
- namespace **decaf::internal::util::concurrent**

7.554 src/main/decaf/internal/util/concurrent/unix/ConditionHandle.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::util::concurrent::ConditionHandle**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.555 src/main/decaf/internal/util/concurrent/windows/ConditionHandle.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::util::concurrent::ConditionHandle**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.556 src/main/decaf/internal/util/concurrent/unix/MutexHandle.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::util::concurrent::MutexHandle**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.557 src/main/decaf/internal/util/concurrent/windows/MutexHandle.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::util::concurrent::MutexHandle**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.558 src/main/decaf/internal/util/HexStringParser.h File Reference

```
#include <decaf/util/Config.h>
```

```
#include <string>
```

Data Structures

- class **decaf::internal::util::HexStringParser**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::internal**
- namespace **decaf::internal::util**

7.559 src/main/decaf/internal/util/TimerTaskHeap.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/TimerTask.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **decaf::internal::util::TimerTaskHeap**
A Binary Heap implemented specifically for the Timer class in Decaf Util.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::internal**
- namespace **decaf::internal::util**

7.560 src/main/decaf/io/BlockingByteArrayInputStream.h File Reference

```
#include <decaf/io/InputStream.h>
#include <decaf/util/concurrent/Mutex.h>
#include <vector>
```

Data Structures

- class **decaf::io::BlockingByteArrayInputStream**
This is a blocking version of a byte buffer stream.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::io**

7.561 src/main/decaf/io/BufferedInputStream.h File Reference

```
#include <decaf/io/FilterInputStream.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
```

Data Structures

- class **decaf::io::BufferedInputStream**

A wrapper around another input stream that performs a buffered read, where it reads more data than it needs in order to reduce the number of io operations on the input stream.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::io**

7.562 src/main/decaf/io/BufferedOutputStream.h File Reference

```
#include <decaf/io/FilterOutputStream.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
```

Data Structures

- class **decaf::io::BufferedOutputStream**

Wrapper around another output stream that buffers output before writing to the target output stream.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::io**

7.563 src/main/decaf/io/ByteArrayInputStream.h File Reference

```
#include <decaf/io/InputStream.h>
#include <decaf/util/concurrent/Mutex.h>
#include <vector>
#include <algorithm>
```

Data Structures

- class **decaf::io::ByteArrayInputStream**

*Simple implementation of **InputStream** (p. 1630) that wraps around an STL Vector `std::vector<unsigned char>`.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::io**

7.564 src/main/decaf/io/ByteArrayOutputStream.h File Reference

```
#include <decaf/io/OutputStream.h>
#include <decaf/util/concurrent/Mutex.h>
#include <vector>
```

Data Structures

- class **decaf::io::ByteArrayOutputStream**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::io**

7.565 src/main/decaf/io/DataInputStream.h File Reference

```
#include <decaf/io/FilterInputStream.h>
#include <decaf/io/IOException.h>
#include <decaf/io/EOFException.h>
#include <decaf/io/UTFDataFormatException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
```

Data Structures

- class **decaf::io::DataInputStream**

A data input stream lets an application read primitive Java data types from an underlying input stream in a machine-independent way.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::io**

7.566 src/main/decaf/io/DataOutputStream.h File Reference

```
#include <decaf/io/FilterOutputStream.h>
#include <decaf/io/IOException.h>
#include <decaf/io/UTFDataFormatException.h>
#include <string>
```

Data Structures

- class **decaf::io::DataOutputStream**

A data output stream lets an application write primitive Java data types to an output stream in a portable way.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::io**

7.567 src/main/decaf/io/EOFException.h File Reference

```
#include <decaf/io/IOException.h>
```

Data Structures

- class **decaf::io::EOFException**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::io**

7.568 src/main/decaf/io/FilterInputStream.h File Reference

```
#include <decaf/io/InputStream.h>
#include <decaf/io/IOException.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
```

Data Structures

- class **decaf::io::FilterInputStream**

*A **FilterInputStream** (p. 1528) contains some other input stream, which it uses as its basic source of data, possibly transforming the data along the way or providing additional functionality.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::io**

7.569 src/main/decaf/io/FilterOutputStream.h File Reference

```
#include <decaf/io/OutputStream.h>
#include <decaf/io/IOException.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/lang/exceptions/NullPointerException.h>
```

Data Structures

- class **decaf::io::FilterOutputStream**

This class is the superclass of all classes that filter output streams.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::io**

7.570 src/main/decaf/io/InputStream.h File Reference

```
#include <decaf/io/IOException.h>
#include <decaf/io/Closeable.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::io::InputStream**

Base interface for an input stream.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::io**

7.571 src/main/decaf/io/InterruptedIOException.h File Reference

```
#include <decaf/lang/Exception.h>
#include <decaf/io/IOException.h>
```

Data Structures

- class **decaf::io::InterruptedIOException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::io**

7.572 src/main/decaf/io/IOException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::io::IOException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::io**

7.573 src/main/decaf/io/OutputStream.h File Reference

```
#include <decaf/io/Closeable.h>
#include <decaf/io/IOException.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/NullPointerException.h>
```

Data Structures

- class **decaf::io::OutputStream**
Base interface for an output stream.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::io**

7.574 src/main/decaf/io/Reader.h File Reference

```
#include <string>
#include <decaf/io/IOException.h>
#include <decaf/io/InputStream.h>
#include <decaf/lang/exceptions/NullPointerException.h>
```

Data Structures

- class **decaf::io::Reader**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::io**

7.575 src/main/decaf/io/UnsupportedEncodingException.h File Reference

```
#include <decaf/io/IOException.h>
```

Data Structures

- class **decaf::io::UnsupportedEncodingException**
Thrown when the the Character Encoding is not supported.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::io**

7.576 src/main/decaf/io/UTFDataFormatException.h File Reference

```
#include <decaf/io/IOException.h>
```

Data Structures

- class **decaf::io::UTFDataFormatException**
Thrown from classes that attempt to read or write a UTF-8 encoded string and an encoding error is encountered.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::io**

7.577 src/main/decaf/io/Writer.h File Reference

```
#include <string>
#include <decaf/io/IOException.h>
#include <decaf/io/OutputStream.h>
#include <decaf/lang/exceptions/NullPointerException.h>
```

Data Structures

- class **decaf::io::Writer**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::io**

7.578 src/main/decaf/lang/Appendable.h File Reference

```
#include <decaf/lang/Exception.h>
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::lang::Appendable**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**

7.579 src/main/decaf/lang/Boolean.h File Reference

```
#include <string>
#include <decaf/lang/Comparable.h>
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::lang::Boolean**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**

7.580 src/main/decaf/lang/Byte.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Number.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/NumberFormatException.h>
#include <string>
```

Data Structures

- class `decaf::lang::Byte`

Namespaces

- namespace `decaf`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `decaf::lang`

7.581 `src/main/decaf/lang/Character.h` File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Number.h>
#include <decaf/lang/Comparable.h>
#include <string>
```

Data Structures

- class `decaf::lang::Character`

Namespaces

- namespace `decaf`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `decaf::lang`

7.582 `src/main/decaf/lang/CharSequence.h` File Reference

```
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/util/Config.h>
```

Data Structures

- class `decaf::lang::CharSequence`

A `CharSequence` (p. 946) is a readable sequence of char values.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**

7.583 src/main/decaf/lang/Comparable.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::lang::Comparable< T >**
This interface imposes a total ordering on the objects of each class that implements it.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**

7.584 src/main/decaf/lang/Double.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/Number.h>
#include <decaf/lang/exceptions/NumberFormatException.h>
#include <string>
```

Data Structures

- class **decaf::lang::Double**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**

7.585 src/main/decaf/lang/Exception.h File Reference

```
#include <decaf/lang/Throwable.h>
#include <decaf/lang/exceptions/ExceptionDefines.h>
#include <decaf/util/Config.h>
#include <stdarg.h>
#include <sstream>
```

Data Structures

- class **decaf::lang::Exception**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**

7.586 src/main/decaf/lang/exceptions/ClassCastException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::lang::exceptions::ClassCastException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

7.587 src/main/decaf/lang/exceptions/IllegalArgumentException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::lang::exceptions::IllegalArgumentException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

7.588 src/main/decaf/lang/exceptions/IllegalMonitorStateException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::lang::exceptions::IllegalMonitorStateException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

7.589 src/main/decaf/lang/exceptions/IllegalThreadStateException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::lang::exceptions::IllegalThreadStateException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

7.590 src/main/decaf/lang/exceptions/IndexOutOfBoundsException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::lang::exceptions::IndexOutOfBoundsException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

7.591 src/main/decaf/lang/exceptions/InterruptedException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::lang::exceptions::InterruptedException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

7.592 src/main/decaf/lang/exceptions/InvalidStateException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::lang::exceptions::InvalidStateException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

7.593 src/main/decaf/lang/exceptions/NoSuchElementException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::lang::exceptions::NoSuchElementException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

7.594 src/main/decaf/lang/exceptions/NullPointerException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::lang::exceptions::NullPointerException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

7.595 src/main/decaf/lang/exceptions/NumberFormatException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::lang::exceptions::NumberFormatException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

7.596 src/main/decaf/lang/exceptions/RuntimeException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::lang::exceptions::RuntimeException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

7.597 src/main/decaf/lang/Float.h File Reference

```
#include <decaf/util/Config.h>  
#include <decaf/lang/Number.h>  
#include <decaf/lang/Comparable.h>
```

```
#include <decaf/lang/exceptions/NumberFormatException.h>
#include <string>
```

Data Structures

- class **decaf::lang::Float**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**

7.598 src/main/decaf/lang/Integer.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Number.h>
#include <decaf/lang/Comparable.h>
#include <string>
#include <decaf/lang/exceptions/NumberFormatException.h>
```

Data Structures

- class **decaf::lang::Integer**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**

7.599 src/main/decaf/lang/Iterable.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/Iterator.h>
```

Data Structures

- class **decaf::lang::Iterable< E >**

*Implementing this interface allows an object to be cast to an **Iterable** (p. 1715) type for generic collections API calls.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**

7.600 src/main/decaf/lang/Long.h File Reference

```
#include <decaf/lang/Number.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/NumberFormatException.h>
#include <string>
```

Data Structures

- class **decaf::lang::Long**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**

7.601 src/main/decaf/lang/Math.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::lang::Math**

*The class **Math** (p. 1999) contains methods for performing basic numeric operations such as the elementary exponential, logarithm, square root, and trigonometric functions.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**

7.602 src/main/decaf/lang/Number.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::lang::Number**

*The abstract class **Number** (p. 2282) is the superclass of classes **Byte** (p. 776), **Double** (p. 1441), **Float** (p. 1544), **Integer** (p. 1652), **Long** (p. 1934), and **Short** (p. 2729).*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**

7.603 src/main/decaf/lang/Pointer.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/ClassCastException.h>
#include <decaf/util/concurrent/atomic/AtomicInteger.h>
#include <decaf/util/Comparator.h>
#include <memory>
#include <typeinfo>
#include <algorithm>
```

Data Structures

- class **decaf::lang::AtomicRefCounter**
- struct **decaf::lang::STATIC_CAST_TOKEN**
- struct **decaf::lang::DYNAMIC_CAST_TOKEN**
- class **decaf::lang::Pointer< T, REFCOUNTER >**

*Decaf's implementation of a Smart **Pointer** (p. 2343) that is a template on a Type and is Thread Safe if the default Reference Counter is used.*

- class **decaf::lang::PointerComparator< T, R >**

*This implementation of `Comparator` is designed to allows objects in a `Collection` to be sorted or tested for equality based on the value of the Object being Pointed to and not the value of the contained pointer in the **`Pointer`** (p. 2343) instance.*

- struct **`std::less< decaf::lang::Pointer< T > >`**

An override of the less function object so that the `Pointer` objects can be stored in STL Maps, etc.

Namespaces

- namespace **`decaf`**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **`decaf::lang`**
- namespace **`std`**

Functions

- template<typename T , typename R , typename U >
bool **`decaf::lang::operator==`** (const `Pointer< T, R >` &left, const U *right)
- template<typename T , typename R , typename U >
bool **`decaf::lang::operator==`** (const U *left, const `Pointer< T, R >` &right)
- template<typename T , typename R , typename U >
bool **`decaf::lang::operator!=`** (const `Pointer< T, R >` &left, const U *right)
- template<typename T , typename R , typename U >
bool **`decaf::lang::operator!=`** (const U *left, const `Pointer< T, R >` &right)

7.604 src/main/decaf/lang/Runnable.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **`decaf::lang::Runnable`**

Interface for a runnable object - defines a task that can be run by a thread.

Namespaces

- namespace **`decaf`**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **`decaf::lang`**

7.605 src/main/decaf/lang/Runtime.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::lang::Runtime**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**

7.606 src/main/decaf/lang/Short.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Number.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/NumberFormatException.h>
#include <string>
```

Data Structures

- class **decaf::lang::Short**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**

7.607 src/main/decaf/lang/System.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/Map.h>
#include <decaf/lang/Exception.h>
#include <decaf/internal/AprPool.h>
#include <string>
```

Data Structures

- class **decaf::lang::System**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**

7.608 src/main/decaf/lang/Thread.h File Reference

```
#include <decaf/lang/exceptions/IllegalThreadStateException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/exceptions/InterruptedException.h>
#include <decaf/lang/exceptions/RuntimeException.h>
#include <decaf/lang/Exception.h>
#include <decaf/lang/Runnable.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/Config.h>
```

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::util::concurrent::locks**
- namespace **decaf::lang**

7.609 src/main/decaf/lang/ThreadGroup.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::lang::ThreadGroup**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**

7.610 src/main/decaf/lang/Throwable.h File Reference

```
#include <string>
#include <vector>
#include <iostream>
#include <exception>
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::lang::Throwable**
This class represents an error that has occurred.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::lang**

7.611 src/main/decaf/net/BindException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/SocketException.h>
```

Data Structures

- class **decaf::net::BindException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::net**

7.612 src/main/decaf/net/BufferedSocket.h File Reference

```
#include <decaf/net/Socket.h>
#include <decaf/net/SocketException.h>
#include <decaf/io/BufferedInputStream.h>
#include <decaf/io/BufferedOutputStream.h>
```

Data Structures

- class **decaf::net::BufferedSocket**
*Buffered **Socket** (p. 2786) class that wraps a **Socket** (p. 2786) derived object and provides Buffered input and Output Streams to improve the efficiency of the reads and writes.*

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::net**

7.613 src/main/decaf/net/ConnectException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/SocketException.h>
```

Data Structures

- class **decaf::net::ConnectException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::net**

7.614 src/main/decaf/net/HttpRetryException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/IOException.h>
```

Data Structures

- class `decaf::net::HttpRetryException`

Namespaces

- namespace `decaf`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `decaf::net`

7.615 src/main/decaf/net/MalformedURLException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/IOException.h>
```

Data Structures

- class `decaf::net::MalformedURLException`

Namespaces

- namespace `decaf`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `decaf::net`

7.616 src/main/decaf/net/NoRouteToHostException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/SocketException.h>
```

Data Structures

- class `decaf::net::NoRouteToHostException`

Namespaces

- namespace `decaf`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::net**

7.617 src/main/decaf/net/PortUnreachableException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/SocketException.h>
```

Data Structures

- class **decaf::net::PortUnreachableException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::net**

7.618 src/main/decaf/net/ProtocolException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/IOException.h>
```

Data Structures

- class **decaf::net::ProtocolException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::net**

7.619 src/main/decaf/net/ServerSocket.h File Reference

```
#include <decaf/net/TcpSocket.h>
#include <decaf/net/SocketException.h>
```



```
#include <decaf/util/Config.h>
#include <decaf/internal/AprPool.h>
#include <apr_network_io.h>
```

Data Structures

- class **decaf::net::ServerSocket**
A server socket class (for testing purposes).

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::net**

7.620 src/main/decaf/net/Socket.h File Reference

```
#include <decaf/net/SocketException.h>
#include <decaf/io/InputStream.h>
#include <decaf/io/OutputStream.h>
#include <decaf/io/Closeable.h>
#include <decaf/util/Config.h>
#include <apr_network_io.h>
```

Data Structures

- class **decaf::net::Socket**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::net**

7.621 src/main/decaf/net/SocketError.h File Reference

```
#include <string>
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::net::SocketError**

Static utility class to simplify handling of error codes for socket operations.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::net**

7.622 src/main/decaf/net/SocketException.h File Reference

```
#include <decaf/io/IOException.h>
```

Data Structures

- class **decaf::net::SocketException**

Exception for errors when manipulating sockets.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::net**

7.623 src/main/decaf/net/SocketFactory.h File Reference

```
#include <decaf/net/SocketException.h>
```

```
#include <decaf/util/Properties.h>
```

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::net::SocketFactory**

Socket (p. 2786) Factory implementation for use in Creating Sockets.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::net**

7.624 src/main/decaf/net/SocketInputStream.h File Reference

```
#include <decaf/io/InputStream.h>
#include <decaf/net/Socket.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/lang/Exception.h>
#include <decaf/lang/exceptions/NullPointerException.h>
```

Data Structures

- class **decaf::net::SocketInputStream**

Input stream for performing reads on a socket.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::net**

7.625 src/main/decaf/net/SocketOutputStream.h File Reference

```
#include <decaf/io/OutputStream.h>
#include <decaf/net/Socket.h>
#include <decaf/util/concurrent/Mutex.h>
```

Data Structures

- class **decaf::net::SocketOutputStream**

Output stream for performing write operations on a socket.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::net**

7.626 src/main/decaf/net/SocketTimeoutException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/InterruptedIOException.h>
```

Data Structures

- class **decaf::net::SocketTimeoutException**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::net**

7.627 src/main/decaf/net/TcpSocket.h File Reference

```
#include <decaf/net/SocketException.h>
#include <decaf/net/Socket.h>
#include <decaf/io/InputStream.h>
#include <decaf/io/OutputStream.h>
#include <decaf/util/Config.h>
#include <decaf/internal/AprPool.h>
```

Data Structures

- class **decaf::net::TcpSocket**

Platform-independent implementation of the socket interface.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::net**

7.628 src/main/decaf/net/UnknownHostException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/IOException.h>
```

Data Structures

- class **decaf::net::UnknownHostException**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::net**

7.629 src/main/decaf/net/UnknownServiceException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/IOException.h>
```

Data Structures

- class **decaf::net::UnknownServiceException**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::net**

7.630 src/main/decaf/net/URI.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/net/URISyntaxException.h>
#include <decaf/net/MalformedURLException.h>
#include <decaf/net/URL.h>
#include <decaf/internal/net/URIType.h>
#include <string>
```

Data Structures

- class **decaf::net::URI**

*This class represents an instance of a **URI** (p. 3096) as defined by RFC 2396.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::net**

7.631 src/main/decaf/net/URISyntaxException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::net::URISyntaxException**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::net**

7.632 src/main/decaf/net/URL.h File Reference

```
#include <decaf/util/Config.h>
#include <string>
```

Data Structures

- class **decaf::net::URL**

*Class **URL** (p. 3133) represents a Uniform Resource Locator, a pointer to a "resource" on the World Wide Web.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::net**

7.633 src/main/decaf/net/URLDecoder.h File Reference

```
#include <decaf/util/Config.h>
#include <string>
```

Data Structures

- class **decaf::net::URLDecoder**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::net**

7.634 src/main/decaf/net/URLEncoder.h File Reference

```
#include <decaf/util/Config.h>
#include <string>
```

Data Structures

- class **decaf::net::URLEncoder**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::net**

7.635 src/main/decaf/nio/Buffer.h File Reference

```
#include <decaf/lang/exceptions/IllegalArgumentException.h>
```

```
#include <decaf/nio/InvalidMarkException.h>
```

Data Structures

- class **decaf::nio::Buffer**

A container for data of a specific primitive type.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::nio**

7.636 src/main/decaf/nio/BufferOverflowException.h File Reference

```
#include <decaf/util/Config.h>
```

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::nio::BufferOverflowException**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::nio**

7.637 src/main/decaf/nio/BufferUnderflowException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::nio::BufferUnderflowException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::nio**

7.638 src/main/decaf/nio/ByteBuffer.h File Reference

```
#include <decaf/nio/Buffer.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
#include <decaf/internal/nio/ByteArrayPerspective.h>
```

Data Structures

- class **decaf::nio::ByteBuffer**
This class defines six categories of operations upon byte buffers:

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::nio**

7.639 src/main/decaf/nio/CharBuffer.h File Reference

```
#include <decaf/nio/Buffer.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
#include <decaf/lang/CharSequence.h>
#include <decaf/lang/Appendable.h>
```

Data Structures

- class **decaf::nio::CharBuffer**

This class defines four categories of operations upon character buffers:

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::nio**

7.640 src/main/decaf/nio/DoubleBuffer.h File Reference

```
#include <decaf/nio/Buffer.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
```

Data Structures

- class **decaf::nio::DoubleBuffer**

This class defines four categories of operations upon double buffers:

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::nio**

7.641 src/main/decaf/nio/FloatBuffer.h File Reference

```
#include <decaf/nio/Buffer.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
```

Data Structures

- class **decaf::nio::FloatBuffer**

This class defines four categories of operations upon float buffers:

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::nio**

7.642 src/main/decaf/nio/IntBuffer.h File Reference

```
#include <decaf/nio/Buffer.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
```

Data Structures

- class **decaf::nio::IntBuffer**

This class defines four categories of operations upon int buffers:

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::nio**

7.643 src/main/decaf/nio/InvalidMarkException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
```

Data Structures

- class **decaf::nio::InvalidMarkException**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::nio**

7.644 src/main/decaf/nio/LongBuffer.h File Reference

```
#include <decaf/nio/Buffer.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
```

Data Structures

- class **decaf::nio::LongBuffer**

This class defines four categories of operations upon long long buffers:

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::nio**

7.645 src/main/decaf/nio/ReadOnlyBufferException.h File Reference

```
#include <decaf/util/Config.h>
```

```
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
```

Data Structures

- class **decaf::nio::ReadOnlyBufferException**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::nio**

7.646 src/main/decaf/nio/ShortBuffer.h File Reference

```
#include <decaf/nio/Buffer.h>
```

```
#include <decaf/lang/Comparable.h>
```

```
#include <decaf/lang/exceptions/NullPointerException.h>
```

```
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
```

```
#include <decaf/nio/BufferUnderflowException.h>
```

```
#include <decaf/nio/BufferOverflowException.h>
```

```
#include <decaf/nio/ReadOnlyBufferException.h>
```

Data Structures

- class `decaf::nio::ShortBuffer`

This class defines four categories of operations upon short buffers:

Namespaces

- namespace `decaf`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `decaf::nio`

7.647 `src/main/decaf/security/auth/x500/X500Principal.h` File Reference

```
#include <string>
#include <vector>
#include <decaf/security/Principal.h>
#include <decaf/util/Map.h>
#include <decaf/io/InputStream.h>
```

Data Structures

- class `decaf::security::auth::x500::X500Principal`

Namespaces

- namespace `decaf`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `decaf::security`
- namespace `decaf::security::auth`
- namespace `decaf::security::auth::x500`

7.648 `src/main/decaf/security/cert/Certificate.h` File Reference

```
#include <vector>
#include <decaf/util/Config.h>
#include <decaf/security/InvalidKeyException.h>
#include <decaf/security/NoSuchAlgorithmException.h>
```

```
#include <decaf/security/SignatureException.h>
#include <decaf/security/cert/CertificateEncodingException.h>
#include <decaf/security/cert/CertificateException.h>
```

Data Structures

- class **decaf::security::cert::Certificate**
Base interface for all identity certificates.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::security**
- namespace **decaf::security::cert**

7.649 src/main/decaf/security/cert/CertificateEncodingException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/security/cert/CertificateException.h>
```

Data Structures

- class **decaf::security::cert::CertificateEncodingException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::security**
- namespace **decaf::security::cert**

7.650 src/main/decaf/security/cert/CertificateException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/security/GeneralSecurityException.h>
```

Data Structures

- class `decaf::security::cert::CertificateException`

Namespaces

- namespace `decaf`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `decaf::security`
- namespace `decaf::security::cert`

7.651 `src/main/decaf/security/cert/CertificateExpiredException.h` File Reference

```
#include <decaf/util/Config.h>
#include <decaf/security/cert/CertificateException.h>
```

Data Structures

- class `decaf::security::cert::CertificateExpiredException`

Namespaces

- namespace `decaf`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `decaf::security`
- namespace `decaf::security::cert`

7.652 `src/main/decaf/security/cert/CertificateNotYetValidException.h` File Reference

```
#include <decaf/util/Config.h>
#include <decaf/security/cert/CertificateException.h>
```

Data Structures

- class `decaf::security::cert::CertificateNotYetValidException`

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::security**
- namespace **decaf::security::cert**

7.653 src/main/decaf/security/cert/CertificateParsingException.h File Reference

```
#include <decaf/util/Config.h>  
#include <decaf/security/cert/CertificateException.h>
```

Data Structures

- class **decaf::security::cert::CertificateParsingException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::security**
- namespace **decaf::security::cert**

7.654 src/main/decaf/security/cert/X509Certificate.h File Reference

```
#include <decaf/security/cert/Certificate.h>  
#include <decaf/util/Config.h>  
#include <decaf/util/Date.h>
```

Data Structures

- class **decaf::security::cert::X509Certificate**
Base interface for all identity certificates.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::security**
- namespace **decaf::security::cert**

7.655 src/main/decaf/security/GeneralSecurityException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::security::GeneralSecurityException**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::security**

7.656 src/main/decaf/security/InvalidKeyException.h File Reference

```
#include <decaf/security/KeyException.h>
```

Data Structures

- class **decaf::security::InvalidKeyException**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::security**

7.657 src/main/decaf/security/Key.h File Reference

```
#include <vector>
```

```
#include <string>
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::security::Key**

*The **Key** (p. 1835) interface is the top-level interface for all keys.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::security**

7.658 src/main/decaf/security/KeyException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/security/GeneralSecurityException.h>
```

Data Structures

- class **decaf::security::KeyException**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::security**

7.659 src/main/decaf/security/NoSuchAlgorithmException.h File Reference

```
#include <decaf/security/GeneralSecurityException.h>
```

Data Structures

- class **decaf::security::NoSuchAlgorithmException**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::security**

7.660 src/main/decaf/security/NoSuchProviderException.h File Reference

```
#include <decaf/util/Config.h>
```

```
#include <decaf/security/GeneralSecurityException.h>
```

Data Structures

- class **decaf::security::NoSuchProviderException**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::security**

7.661 src/main/decaf/security/Principal.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::security::Principal**

Base interface for a principal, which can represent an individual or organization.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::security**

7.662 src/main/decaf/security/PublicKey.h File Reference

```
#include <decaf/security/Key.h>
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::security::PublicKey**
A public key.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::security**

7.663 src/main/decaf/security/SignatureException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/security/GeneralSecurityException.h>
```

Data Structures

- class **decaf::security::SignatureException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::security**

7.664 src/main/decaf/security_provider/SecurityProvider.h File Reference

Data Structures

- class **decaf::security_provider::SecurityProvider**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::security_provider**

7.665 src/main/decaf/security_provider/SecurityProviderMap.h File Reference

```
#include <map>
#include <vector>
#include <string>
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::security_provider::SecurityProviderMap**

Lookup Map for Connector Factories.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::security_provider**

7.666 src/main/decaf/security_provider/SecurityProviderRegistrar.h File Reference

```
#include <string>
#include <decaf/security_provider/SecurityProviderMap.h>
```

Data Structures

- class **decaf::security_provider::SecurityProviderRegistrar**

Registers the passed in provider into the provider map, this class can manage the lifetime of the registered provider (default behaviour).

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::security_provider**

7.667 src/main/decaf/security_provider/unix/openssl/OpenSSLX500Principal.h File Reference

```
#include <decaf/security/auth/x500/X500Principal.h>  
#include <openssl/x509.h>
```

Data Structures

- class **decaf::security_provider::unix::openssl::OpenSSLX500Principal**
The `OpenSSLX500Principal` (p. 2287) wraps around an `OpenSSL X509_NAME` structure.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::security_provider**
- namespace **decaf::security_provider::unix**
- namespace **decaf::security_provider::unix::openssl**

7.668 src/main/decaf/security_provider/unix/openssl/OpenSSLX509Certificate.h File Reference

```
#include <decaf/security/cert/X509Certificate.h>
```

Data Structures

- class **decaf::security_provider::unix::openssl::OpenSSLX509Certificate**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::security_provider**

- namespace `decaf::security_provider::unix`
- namespace `decaf::security_provider::unix::openssl`

7.669 `src/main/decaf/util/AbstractCollection.h` File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/Iterable.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/Collection.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/concurrent/Mutex.h>
#include <memory>
```

Data Structures

- class `decaf::util::AbstractCollection< E >`

*This class provides a skeletal implementation of the **Collection** (p. 982) interface, to minimize the effort required to implement this interface.*

Namespaces

- namespace `decaf`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `decaf::util`

7.670 `src/main/decaf/util/AbstractList.h` File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/Iterable.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/List.h>
```



```
#include <memory>
```

Data Structures

- class `decaf::util::AbstractList< E >`

*This class provides a skeletal implementation of the **List** (p. 1865) interface to minimize the effort required to implement this interface backed by a "random access" data store (such as an array).*

Namespaces

- namespace `decaf`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `decaf::util`

7.671 src/main/decaf/util/AbstractMap.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/Map.h>
#include <decaf/util/Set.h>
#include <memory>
```

Data Structures

- class `decaf::util::AbstractMap< K, V, COMPARATOR >`

*This class provides a skeletal implementation of the **Map** (p. 1970) interface, to minimize the effort required to implement this interface.*

Namespaces

- namespace `decaf`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `decaf::util`

7.672 src/main/decaf/util/AbstractQueue.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
#include <decaf/lang/Iterable.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/Queue.h>
#include <memory>
```

Data Structures

- class **decaf::util::AbstractQueue**< **E** >

*This class provides skeletal implementations of some **Queue** (p. 2507) operations.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**

7.673 src/main/decaf/util/AbstractSequentialList.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/Iterable.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/AbstractList.h>
#include <memory>
```

Data Structures

- class **decaf::util::AbstractSequentialList**< **E** >

*This class provides a skeletal implementation of the **List** (p. 1865) interface to minimize the effort required to implement this interface backed by a "sequential access" data store (such as a linked list).*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**

7.674 src/main/decaf/util/AbstractSet.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/Iterable.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/Set.h>
#include <memory>
```

Data Structures

- class **decaf::util::AbstractSet< E >**

*This class provides a skeletal implementation of the **Set** (p. 2729) interface to minimize the effort required to implement this interface.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**

7.675 src/main/decaf/util/Collection.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
```

```
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/Iterable.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/concurrent/Synchronizable.h>
```

Data Structures

- class **decaf::util::Collection**< **E** >
The root interface in the collection hierarchy.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**

7.676 src/main/decaf/util/Comparator.h File Reference

```
#include <decaf/util/Config.h>
#include <algorithm>
#include <functional>
```

Data Structures

- class **decaf::util::Comparator**< **T** >
A comparison function, which imposes a total ordering on some collection of objects.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**

7.677 src/main/decaf/util/comparators/Less.h File Reference

```
#include <decaf/util/Comparator.h>
```

Data Structures

- class **decaf::util::comparators::Less**< E >

*Simple **Less** (p. 1862) **Comparator** (p. 1011) that compares to elements to determine if the first is less than the second.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::comparators**

7.678 src/main/decaf/util/concurrent/atomic/AtomicBoolean.h File Reference

```
#include <string>
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::util::concurrent::atomic::AtomicBoolean**

A boolean value that may be updated atomically.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::util::concurrent::atomic**

7.679 src/main/decaf/util/concurrent/atomic/AtomicInteger.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Number.h>
#include <string>
```

Data Structures

- class **decaf::util::concurrent::atomic::AtomicInteger**

An int value that may be updated atomically.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::util::concurrent::atomic**

7.680 src/main/decaf/util/concurrent/atomic/AtomicReference.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Long.h>
#include <apr_atomic.h>
```

Data Structures

- class **decaf::util::concurrent::atomic::AtomicReference< T >**

An Pointer reference that may be updated atomically.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::util::concurrent::atomic**

7.681 src/main/decaf/util/concurrent/BlockingQueue.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/AbstractQueue.h>
#include <decaf/util/concurrent/TimeUnit.h>
#include <decaf/lang/exceptions/InterruptedException.h>
```

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.682 src/main/decaf/util/concurrent/BrokenBarrierException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::util::concurrent::BrokenBarrierException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.683 src/main/decaf/util/concurrent/Callable.h File Reference

```
#include <decaf/util/Config.h>  
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::util::concurrent::Callable< V >**
A task that returns a result and may throw an exception.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.684 src/main/decaf/util/concurrent/CancellationException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::util::concurrent::CancellationException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.685 src/main/decaf/util/concurrent/Concurrent.h File Reference

```
#include <decaf/util/concurrent/Lock.h>
```

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

Defines

- **#define WAIT_INFINITE 0xFFFFFFFF**
The synchronized macro defines a mechanism for synchronizing a section of code.
- **#define synchronized(W)**

7.685.1 Define Documentation

7.685.1.1 #define synchronized(W)

Value:

```
if(false){}
    else
        for( decaf::util::concurrent::Lock lock_W(W);
            lock_W.isLocked(); lock_W.unlock() )
```


7.685.1.2 `#define WAIT_INFINITE 0xFFFFFFFF`

The synchronized macro defines a mechanism for synchronizing a section of code.

The macro must be passed an object that implements the Synchronizable interface.

The macro works by creating a for loop that will loop exactly once, creating a Lock object that is scoped to the loop. Once the loop completes and exits the Lock object goes out of scope releasing the lock on object W. For added safety the if else is used because not all compilers restrict the lifetime of loop variables to the loop, they will however restrict them to the scope of the else.

The macro would be used as follows.

Synchronizable X;

```
somefunction() { synchronized(X) (p. 3620) { // Do something that needs synchronizing. } }
```

7.686 src/main/decaf/util/concurrent/ConcurrentMap.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/Map.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
```

Data Structures

- class `decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR >`
*Interface for a **Map** (p. 1970) type that provides additional atomic `putIfAbsent`, `remove`, and `replace` methods alongside the already available **Map** (p. 1970) interface.*

Namespaces

- namespace `decaf`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `decaf::util`
- namespace `decaf::util::concurrent`

7.687 src/main/decaf/util/concurrent/ConcurrentStlMap.h File Reference

```
#include <map>
#include <vector>
#include <decaf/lang/exceptions/NoSuchElementException.h>
```

```
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/concurrent/ConcurrentMap.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/Map.h>
```

Data Structures

- class **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >**
Map (p. 1970) template that wraps around a `std::map` to provide a more user-friendly interface and to provide common functions that do not exist in `std::map`.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.688 src/main/decaf/util/concurrent/CountDownLatch.h File Reference

```
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/Config.h>
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::util::concurrent::CountDownLatch**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.689 src/main/decaf/util/concurrent/Delayed.h File Reference

```
#include <decaf/util/Config.h>
```

```
#include <decaf/lang/Comparable.h>
#include <decaf/util/concurrent/TimeUnit.h>
```

Data Structures

- class **decaf::util::concurrent::Delayed**
A mix-in style interface for marking objects that should be acted upon after a given delay.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.690 src/main/decaf/util/concurrent/ExecutionException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::util::concurrent::ExecutionException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.691 src/main/decaf/util/concurrent/Executor.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Runnable.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/util/concurrent/RejectedExecutionException.h>
```

Data Structures

- class **decaf::util::concurrent::Executor**
*An object that executes submitted **decaf.lang Runnable** (p. 2642) tasks.*

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.692 src/main/decaf/util/concurrent/ExecutorService.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/concurrent/Executor.h>
#include <decaf/util/concurrent/TimeUnit.h>
#include <decaf/lang/exceptions/InterruptedException.h>
```

Data Structures

- class **decaf::util::concurrent::ExecutorService**
*An **Executor** (p. 1509) that provides methods to manage termination and methods that can produce a **Future** (p. 1597) for tracking progress of one or more asynchronous tasks.*

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.693 src/main/decaf/util/concurrent/Future.h File Reference

Data Structures

- class **decaf::util::concurrent::Future< V >**
*A **Future** (p. 1597) represents the result of an asynchronous computation.*

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.694 src/main/decaf/util/concurrent/Lock.h File Reference

```
#include <decaf/lang/Exception.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::util::concurrent::Lock**
A wrapper class around a given synchronization mechanism that provides automatic release upon destruction.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.695 src/main/decaf/util/concurrent/locks/Lock.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/InterruptedException.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/util/concurrent/locks/Condition.h>
```

Data Structures

- class **decaf::util::concurrent::locks::Lock**
***Lock** (p. 1898) implementations provide more extensive locking operations than can be obtained using synchronized statements.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::util::concurrent::locks**

7.696 src/main/decaf/util/concurrent/locks/Condition.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/Date.h>
#include <decaf/util/concurrent/TimeUnit.h>
#include <decaf/lang/exceptions/RuntimeException.h>
#include <decaf/lang/exceptions/InterruptedException.h>
#include <decaf/lang/exceptions/IllegalMonitorStateException.h>
```

Data Structures

- class **decaf::util::concurrent::locks::Condition**

***Condition** (p. 1040) factors out the **Mutex** (p. 2239) monitor methods (*wait*, *notify* and *notifyAll*) into distinct objects to give the effect of having multiple wait-sets per object, by combining them with the use of arbitrary **Lock** (p. 1898) implementations.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::util::concurrent::locks**

7.697 src/main/decaf/util/concurrent/locks/LockSupport.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::util::concurrent::locks::LockSupport**

Basic thread blocking primitives for creating locks and other synchronization classes.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**
- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::util::concurrent::locks**

7.698 src/main/decaf/util/concurrent/locks/ReadWriteLock.h File Reference

Data Structures

- class **decaf::util::concurrent::locks::ReadWriteLock**

*A **ReadWriteLock** (p. 2522) maintains a pair of associated locks, one for read-only operations and one for writing.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::util::concurrent::locks**

7.699 src/main/decaf/util/concurrent/locks/ReentrantLock.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/concurrent/locks/Lock.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **decaf::util::concurrent::locks::ReentrantLock**

*A reentrant mutual exclusion **Lock** (p. 1898) with extended capabilities.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::util::concurrent::locks**

7.700 src/main/decaf/util/concurrent/Mutex.h File Reference

```
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/concurrent/Concurrent.h>
#include <decaf/lang/Thread.h>
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::util::concurrent::Mutex**

***Mutex** (p. 2239) object that offers recursive support on all platforms as well as providing the ability to use the standard wait / notify pattern used in languages like Java.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.701 src/main/decaf/util/concurrent/PooledThread.h File Reference

```
#include <decaf/lang/Thread.h>
#include <decaf/util/concurrent/PooledThreadListener.h>
```



```
#include <decaf/util/logging/LoggerDefines.h>
#include <decaf/util/Config.h>
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::util::concurrent::PooledThread**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.702 src/main/decaf/util/concurrent/PooledThreadListener.h File Reference

```
#include <decaf/lang/Exception.h>
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::util::concurrent::PooledThreadListener**
Abstract Listener Interface for users of `ThreadPool` (p. 2977).

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.703 src/main/decaf/util/concurrent/RejectedExecutionException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::util::concurrent::RejectedExecutionException**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.704 src/main/decaf/util/concurrent/RejectedExecutionHandler.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Runnable.h>
#include <decaf/util/concurrent/RejectedExecutionException.h>
```

Data Structures

- class **decaf::util::concurrent::RejectedExecutionHandler**
*A handler for tasks that cannot be executed by a **ThreadPoolExecutor** (p.??).*

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.705 src/main/decaf/util/concurrent/Semaphore.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/InterruptedException.h>
#include <decaf/lang/exceptions/RuntimeException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/util/concurrent/TimeUnit.h>
#include <memory>
```

Data Structures

- class **decaf::util::concurrent::Semaphore**

A counting semaphore.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.706 src/main/decaf/util/concurrent/Synchronizable.h File Reference

```
#include <decaf/lang/exceptions/RuntimeException.h>
#include <decaf/lang/exceptions/InterruptedException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/exceptions/IllegalMonitorStateException.h>
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::util::concurrent::Synchronizable**
The interface for all synchronizable objects (that is, objects that can be locked and unlocked).

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.707 src/main/decaf/util/concurrent/SynchronousQueue.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/concurrent/BlockingQueue.h>
#include <vector>
```

Data Structures

- class `decaf::util::concurrent::SynchronousQueue< E >`
A `BlockingQueue` blocking queue} in which each insert operation must wait for a corresponding remove operation by another thread, and vice versa.
- class `decaf::util::concurrent::SynchronousQueue< E >::EmptyIterator`

Namespaces

- namespace `decaf`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `decaf::util`
- namespace `decaf::util::concurrent`

7.708 `src/main/decaf/util/concurrent/TaskListener.h` File Reference

```
#include <decaf/lang/Runnable.h>
#include <decaf/lang/Exception.h>
#include <decaf/util/Config.h>
```

Data Structures

- class `decaf::util::concurrent::TaskListener`

Namespaces

- namespace `decaf`
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace `decaf::util`
- namespace `decaf::util::concurrent`

7.709 `src/main/decaf/util/concurrent/ThreadFactory.h` File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class `decaf::util::concurrent::ThreadFactory`
*public interface **ThreadFactory** (p. 2976)*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.710 src/main/decaf/util/concurrent/ThreadPool.h File Reference

```
#include <decaf/lang/Runnable.h>
#include <decaf/util/concurrent/PooledThread.h>
#include <decaf/util/concurrent/PooledThreadListener.h>
#include <decaf/util/concurrent/TaskListener.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/StlQueue.h>
#include <decaf/util/logging/LoggerDefines.h>
#include <decaf/util/Config.h>
#include <vector>
```

Data Structures

- class **decaf::util::concurrent::ThreadPool**

Defines a Thread Pool object that implements the functionality of pooling threads to perform user tasks.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.711 src/main/decaf/util/concurrent/TimeoutException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class `decaf::util::concurrent::TimeoutException`

Namespaces

- namespace `decaf`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `decaf::util`
- namespace `decaf::util::concurrent`

7.712 `src/main/decaf/util/concurrent/TimeUnit.h` File Reference

```
#include <string>
#include <decaf/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/exceptions/InterruptedException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
```

Data Structures

- class `decaf::util::concurrent::TimeUnit`

*A **TimeUnit** (p. 3005) represents time durations at a given unit of granularity and provides utility methods to convert across units, and to perform timing and delay operations in these units.*

Namespaces

- namespace `decaf`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `decaf::lang`
- namespace `decaf::util`
- namespace `decaf::util::concurrent`

7.713 src/main/decaf/util/Date.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <string>
```

Data Structures

- class **decaf::util::Date**
Wrapper class around a time value in milliseconds.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**

7.714 src/main/decaf/util/Iterator.h File Reference

```
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
```

Data Structures

- class **decaf::util::Iterator< T >**
Defines an object that can be used to iterate over the elements of a collection.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**

7.715 src/main/decaf/util/List.h File Reference

```
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
```

```
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/Config.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/AbstractCollection.h>
#include <decaf/util/ListIterator.h>
```

Data Structures

- class **decaf::util::List**< E >
An ordered collection (also known as a sequence).

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**

7.716 src/main/decaf/util/ListIterator.h File Reference

```
#include <decaf/util/Iterator.h>
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
```

Data Structures

- class **decaf::util::ListIterator**< E >
An iterator for lists that allows the programmer to traverse the list in either direction, modify the list during iteration, and obtain the iterator's current position in the list.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**

7.717 src/main/decaf/util/logging/ConsoleHandler.h File Reference

```
#include <decaf/util/logging/StreamHandler.h>
```



```
#include <decaf/io/StandardErrorOutputStream.h>
```

7.718 src/main/decaf/util/logging/Filter.h File Reference

```
#include <decaf/util/logging/LogRecord.h>
```

Data Structures

- class **decaf::util::logging::Filter**

*A **Filter** (p. 1527) can be used to provide fine grain control over what is logged, beyond the control provided by log levels.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::logging**

7.719 src/main/decaf/util/logging/Formatter.h File Reference

Data Structures

- class **decaf::util::logging::Formatter**

*A **Formatter** (p. 1595) provides support for formatting LogRecords.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::logging**

7.720 src/main/decaf/util/logging/Handler.h File Reference

```
#include <decaf/io/Closeable.h>
```

```
#include <decaf/util/logging/LogRecord.h>
```

Data Structures

- class **decaf::util::logging::Handler**

A **Handler** (p. 1604) object takes log messages from a **Logger** (p. 1908) and exports them.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::logging**

7.721 src/main/decaf/util/logging/Logger.h File Reference

```
#include <decaf/util/logging/LoggerCommon.h>
#include <decaf/util/logging/LogRecord.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/util/Config.h>
#include <list>
#include <string>
#include <stdarg.h>
```

Data Structures

- class **decaf::util::logging::Logger**

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::logging**

7.722 src/main/decaf/util/logging/LoggerCommon.h File Reference

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::logging**

Enumerations

- enum **decaf::util::logging::Level** {
decaf::util::logging::Off, **decaf::util::logging::Null**, **decaf::util::logging::Markblock**,
decaf::util::logging::Debug,
decaf::util::logging::Info, **decaf::util::logging::Warn**, **decaf::util::logging::Error**,
decaf::util::logging::Fatal,
decaf::util::logging::Throwing }

Defines an enumeration for logging levels.

7.723 src/main/decaf/util/logging/LoggerDefines.h File Reference

```
#include <decaf/util/logging/SimpleLogger.h>
#include <sstream>
```

Defines

- **#define LOGDECAF_DECLARE(loggerName) static decaf::util::logging::SimpleLogger loggerName;**
- **#define LOGDECAF_INITIALIZE(loggerName, className, loggerFamily) decaf::util::logging::SimpleLogger className::loggerName(loggerFamily);**
- **#define LOGDECAF_DECLARE_LOCAL(loggerName) decaf::util::logging::Logger loggerName;**
- **#define LOGDECAF_DEBUG(logger, message) logger.debug(__FILE__, __LINE__, message);**
- **#define LOGDECAF_DEBUG_1(logger, message, value)**
- **#define LOGDECAF_INFO(logger, message) logger.info(__FILE__, __LINE__, message);**
- **#define LOGDECAF_ERROR(logger, message) logger.error(__FILE__, __LINE__, message);**
- **#define LOGDECAF_WARN(logger, message) logger.warn(__FILE__, __LINE__, message);**
- **#define LOGDECAF_FATAL(logger, message) logger.fatal(__FILE__, __LINE__, message);**

7.723.1 Define Documentation

7.723.1.1 `#define LOGDECAF_DEBUG(logger, message)`
`logger.debug(__FILE__, __LINE__, message);`

7.723.1.2 `#define LOGDECAF_DEBUG_1(logger, message, value)`

Value:

```

;          \
{          \
    std::ostream ostream;          \
    ostream << message << value;    \
    logger.debug(__FILE__, __LINE__, ostream.str()); \
}

```

7.723.1.3 `#define LOGDECAF_DECLARE(loggerName)` static
`decaf::util::logging::SimpleLogger loggerName;`

7.723.1.4 `#define LOGDECAF_DECLARE_LOCAL(loggerName)`
`decaf::util::logging::Logger loggerName;`

7.723.1.5 `#define LOGDECAF_ERROR(logger, message)`
`logger.error(__FILE__, __LINE__, message);`

7.723.1.6 `#define LOGDECAF_FATAL(logger, message)`
`logger.fatal(__FILE__, __LINE__, message);`

7.723.1.7 `#define LOGDECAF_INFO(logger, message)`
`logger.info(__FILE__, __LINE__, message);`

7.723.1.8 `#define LOGDECAF_INITIALIZE(loggerName,
className, loggerFamily)` `decaf::util::logging::SimpleLogger`
`className::loggerName(loggerFamily);`

7.723.1.9 `#define LOGDECAF_WARN(logger, message)`
`logger.warn(__FILE__, __LINE__, message);`

7.724 `src/main/decaf/util/logging/LoggerHierarchy.h` File Reference

Data Structures

- class `decaf::util::logging::LoggerHierarchy`

Namespaces

- namespace `decaf`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::logging**

7.725 src/main/decaf/util/logging/LogManager.h File Reference

```
#include <map>
#include <list>
#include <string>
#include <vector>
#include <decaf/util/Properties.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::util::logging::LogManager**

*There is a single global **LogManager** (p. 1923) object that is used to maintain a set of shared state about Loggers and log services.*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::logging**

7.726 src/main/decaf/util/logging/LogRecord.h File Reference

```
#include <decaf/util/logging/LoggerCommon.h>
#include <decaf/util/Config.h>
#include <string>
```

Data Structures

- class **decaf::util::logging::LogRecord**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::logging**

7.727 src/main/decaf/util/logging/LogWriter.h File Reference

```
#include <decaf/util/concurrent/Mutex.h>
```

Data Structures

- class **decaf::util::logging::LogWriter**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::logging**

7.728 src/main/decaf/util/logging/MarkBlockLogger.h File Reference

```
#include <decaf/util/logging/Logger.h>
```

Data Structures

- class **decaf::util::logging::MarkBlockLogger**
Defines a class that can be used to mark the entry and exit from scoped blocks.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::logging**

7.729 src/main/decaf/util/logging/PropertiesChangeListener.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::util::logging::PropertiesChangeListener**
*Defines the interface that classes can use to listen for change events on **Properties** (p. 2494).*

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::logging**

7.730 src/main/decaf/util/logging/SimpleFormatter.h File Reference

```
#include <decaf/util/logging/formatter.h>
```

Data Structures

- class **decaf::util::logging::SimpleFormatter**
*Print a brief summary of the **LogRecord** (p. 1928) in a human readable format.*

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::logging**

7.731 src/main/decaf/util/logging/SimpleLogger.h File Reference

```
#include <string>
```

```
#include <decaf/util/Config.h>
```

Data Structures

- class `decaf::util::logging::SimpleLogger`

Namespaces

- namespace `decaf`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `decaf::util`
- namespace `decaf::util::logging`

7.732 `src/main/decaf/util/logging/StreamHandler.h` File Reference

```
#include <decaf/util/logging/LoggerCommon.h>
#include <decaf/util/logging/Handler.h>
#include <decaf/util/logging/Formatter.h>
#include <decaf/util/logging/Filter.h>
#include <decaf/io/OutputStream.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/InvalidStateException.h>
#include <decaf/util/concurrent/Concurrent.h>
#include <decaf/util/Config.h>
```

Data Structures

- class `decaf::util::logging::StreamHandler`

Namespaces

- namespace `decaf`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `decaf::util`
- namespace `decaf::util::logging`

7.733 `src/main/decaf/util/Map.h` File Reference

```
#include <functional>
#include <vector>
```



```
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/concurrent/Mutex.h>
```

Data Structures

- class **decaf::util::Map**< K, V, COMPARATOR >
Map (p. 1970) template that wraps around a `std::map` to provide a more user-friendly interface and to provide common functions that do not exist in `std::map`.
- class **decaf::util::Map**< K, V, COMPARATOR >::Entry

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**

7.734 src/main/decaf/util/PriorityQueue.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/Collection.h>
#include <decaf/util/AbstractQueue.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/Comparator.h>
#include <decaf/util/comparators/Less.h>
#include <decaf/lang/Math.h>
#include <decaf/lang/Pointer.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <memory>
```

Data Structures

- class **decaf::util::PriorityQueue**< E >
An unbounded priority queue based on a binary heap algorithm.
- class **decaf::util::PriorityQueue**< E >::PriorityQueueIterator
- class **decaf::util::PriorityQueue**< E >::ConstPriorityQueueIterator

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**

7.735 src/main/decaf/util/Properties.h File Reference

```
#include <memory>
#include <vector>
#include <string>
#include <decaf/util/Config.h>
#include <decaf/io/InputStream.h>
#include <decaf/io/OutputStream.h>
#include <decaf/io/Reader.h>
#include <decaf/io/Writer.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/io/IOException.h>
```

Data Structures

- class **decaf::util::Properties**

Java-like properties class for mapping string names to string values.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**

7.736 src/main/decaf/util/Random.h File Reference

```
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/util/Config.h>
#include <vector>
#include <cmath>
```

Data Structures

- class **decaf::util::Random**
Random (p. 2513) Value Generator which is used to generate a stream of pseudorandom numbers.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**

7.737 src/main/decaf/util/Set.h File Reference

```
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/AbstractCollection.h>
```

Data Structures

- class **decaf::util::Set< E >**
A collection that contains no duplicate elements.

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**

7.738 src/main/decaf/util/StlList.h File Reference

```
#include <list>
#include <algorithm>
#include <memory>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NoSuchElementException>
```

```
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/Config.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/ListIterator.h>
#include <decaf/util/List.h>
```

Data Structures

- class **decaf::util::StlList< E >**
List (p. 1865) class that wraps the STL list object to provide a simpler interface and additional methods not provided by the STL type.
- class **decaf::util::StlList< E >::StlListIterator**
- class **decaf::util::StlList< E >::ConstStlListIterator**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**

7.739 src/main/decaf/util/StlMap.h File Reference

```
#include <map>
#include <vector>
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/Map.h>
```

Data Structures

- class **decaf::util::StlMap< K, V, COMPARATOR >**
Map (p. 1970) template that wraps around a `std::map` to provide a more user-friendly interface and to provide common functions that do not exist in `std::map`.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**

7.740 src/main/decaf/util/StlQueue.h File Reference

```
#include <list>
#include <vector>
#include <decaf/util/Iterator.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::util::StlQueue< T >**
*The **Queue** (p. 2507) class accepts messages with an **psuh(m)** command where **m** is the message to be queued.*
- class **decaf::util::StlQueue< T >::QueueIterator**

Namespaces

- namespace **decaf**
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**

7.741 src/main/decaf/util/StlSet.h File Reference

```
#include <set>
#include <vector>
#include <memory>
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/AbstractSet.h>
```

Data Structures

- class **decaf::util::StlSet< E >**

Set (p. 2729) template that wraps around a `std::set` to provide a more user-friendly interface and to provide common functions that do not exist in `std::set`.

- class `decaf::util::StlSet< E >::SetIterator`
- class `decaf::util::StlSet< E >::ConstSetIterator`

Namespaces

- namespace `decaf`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `decaf::util`

7.742 src/main/decaf/util/StringTokenizer.h File Reference

```
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/util/Config.h>
#include <string>
```

Data Structures

- class `decaf::util::StringTokenizer`

Namespaces

- namespace `decaf`

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace `decaf::util`

7.743 src/main/decaf/util/Timer.h File Reference

```
#include <memory>
#include <decaf/util/Config.h>
#include <decaf/util/Date.h>
#include <decaf/lang/Pointer.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
```

Data Structures

- class **decaf::util::Timer**

A facility for threads to schedule tasks for future execution in a background thread.

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**

7.744 src/main/decaf/util/TimerTask.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Runnable.h>
#include <decaf/util/concurrent/Mutex.h>
```

Data Structures

- class **decaf::util::TimerTask**

*A Base class for a task object that can be scheduled for one-time or repeated execution by a **Timer** (p. 2989).*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::internal**
- namespace **decaf::internal::util**
- namespace **decaf::util**

7.745 src/main/decaf/util/UUID.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <apr_uuid.h>
#include <string>
```

Data Structures

- class **decaf::util::UUID**

*A class that represents an immutable universally unique identifier (**UUID** (p. 3141)).*

Namespaces

- namespace **decaf**

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**

Index

- ~AbstractCollection
 - decaf::util::AbstractCollection, 122
- ~AbstractList
 - decaf::util::AbstractList, 132
- ~AbstractMap
 - decaf::util::AbstractMap, 133
- ~AbstractQueue
 - decaf::util::AbstractQueue, 135
- ~AbstractSequentialList
 - decaf::util::AbstractSequentialList, 138
- ~AbstractSet
 - decaf::util::AbstractSet, 139
- ~AbstractTransportFactory
 - activemq::transport::AbstractTransportFactory, 140
- ~ActiveMQAckHandler
 - activemq::core::ActiveMQAckHandler, 141
- ~ActiveMQBlobMessage
 - activemq::commands::ActiveMQBlobMessage, 143
- ~ActiveMQBlobMessageMarshaller
 - activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller, 152
 - activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller, 164
 - activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller, 148
 - activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller, 156
 - activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller, 160
- ~ActiveMQBytesMessage
 - activemq::commands::ActiveMQBytesMessage, 170
- ~ActiveMQBytesMessageMarshaller
 - activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller, 187
 - activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller, 199
 - activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller, 183
 - activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller, 191
 - activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller, 195
- ~ActiveMQCPP
 - activemq::library::ActiveMQCPP, 239
- ~ActiveMQConnection
 - activemq::core::ActiveMQConnection, 205
- ~ActiveMQConnectionFactory
 - activemq::core::ActiveMQConnectionFactory, 213
- ~ActiveMQConnectionMetaData
 - activemq::core::ActiveMQConnectionMetaData, 217
- ~ActiveMQConnectionSupport
 - activemq::core::ActiveMQConnectionSupport, 222
- ~ActiveMQConsumer
 - activemq::core::ActiveMQConsumer, 232
- ~ActiveMQDestination
 - activemq::commands::ActiveMQDestination, 242
- ~ActiveMQDestinationMarshaller
 - activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller, 255
 - activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller, 267
 - activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller, 251
 - activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller, 259
 - activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller, 263
- ~ActiveMQException
 - activemq::exceptions::ActiveMQException, 271
- ~ActiveMQMapMessage
 - activemq::commands::ActiveMQMapMessage, 275
- ~ActiveMQMapMessageMarshaller
 - activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller, 287
 - activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller, 299
 - activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller, 283
 - activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller, 291
 - activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller, 295

- 298
- ~ActiveMQMessage
 - activemq::commands::ActiveMQMessage, 305
- ~ActiveMQMessageMarshaller
 - activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller, 312
 - activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller, 324
 - activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller, 308
 - activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller, 316
 - activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller, 320
- ~ActiveMQMessageTemplate
 - activemq::commands::ActiveMQMessageTemplate, 331
- ~ActiveMQObjectMessage
 - activemq::commands::ActiveMQObjectMessage, 345
- ~ActiveMQObjectMessageMarshaller
 - activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller, 352
 - activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller, 364
 - activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller, 348
 - activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller, 356
 - activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller, 360
- ~ActiveMQProducer
 - activemq::core::ActiveMQProducer, 369
- ~ActiveMQProperties
 - activemq::util::ActiveMQProperties, 376
- ~ActiveMQQueue
 - activemq::commands::ActiveMQQueue, 380
- ~ActiveMQQueueMarshaller
 - activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller, 388
 - activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller, 400
 - activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller, 384
 - activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller, 392
 - activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller, 396
- ~ActiveMQSession
 - activemq::core::ActiveMQSession, 406
- ~ActiveMQSessionExecutor
 - activemq::core::ActiveMQSessionExecutor, 419
- ~ActiveMQStreamMessage
 - activemq::commands::ActiveMQStreamMessage, 425
- ~ActiveMQStreamMessageMarshaller
 - activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller, 441
 - activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller, 453
 - activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller, 437
 - activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller, 445
 - activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller, 449
- ~ActiveMQTempDestination
 - activemq::commands::ActiveMQTempDestination, 457
- ~ActiveMQTempDestinationMarshaller
 - activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller, 463
 - activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller, 474
 - activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller, 460
 - activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller, 467
 - activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller, 471
- ~ActiveMQTempQueue
 - activemq::commands::ActiveMQTempQueue, 478
- ~ActiveMQTempQueueMarshaller
 - activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller, 486
 - activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller, 498
 - activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller, 482
 - activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller, 490
 - activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller, 494
- ~ActiveMQTempTopic
 - activemq::commands::ActiveMQTempTopic, 502
- ~ActiveMQTempTopicMarshaller
 - activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller, 522
 - activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller, 506
 - activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller, 514
 - activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller, 522
 - activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller, 510

activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller, 598	~BaseCommandMarshaller
514	
activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller, 611	
518	
~ActiveMQTextMessage	activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller, 631
activemq::commands::ActiveMQTextMessage, 526	activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller, 604
~ActiveMQTextMessageMarshaller	
activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller, 535	activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller, 618
activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller, 547	activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller, 624
activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller, 531	~BaseDataStreamMarshaller, 639
activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller, 539	~BaseDataStructure
activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller, 543	activemq::wireformat::openwire::marshal::v1::BaseDataStructure, 660
~ActiveMQTopic	~BindException
activemq::commands::ActiveMQTopic, 551	decaf::net::BindException, 665
~ActiveMQTopicMarshaller	~BlockingByteArrayInputStream
activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller, 558	decaf::net::BlockingByteArrayInputStream, 668
activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller, 570	~Boolean
activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller, 554	decaf::lang::Boolean, 675
activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller, 562	activemq::commands::BooleanExpression, 679
activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller, 566	~BooleanStream
~ActiveMQTransactionContext	activemq::wireformat::openwire::utils::BooleanStream, 682
activemq::core::ActiveMQTransactionContext, 574	~BrokenBarrierException
~Appendable	decaf::util::concurrent::BrokenBarrierException, 685
decaf::lang::Appendable, 577	~BrokerError
~AprPool	activemq::commands::BrokerError, 687
decaf::internal::AprPool, 579	~BrokerException
~AtomicBoolean	activemq::exceptions::BrokerException, 690
decaf::util::concurrent::atomic::AtomicBoolean, 580	~BrokerId
~AtomicInteger	activemq::commands::BrokerId, 692
decaf::util::concurrent::atomic::AtomicInteger, 583	~BrokerIdMarshaller
~AtomicReference	activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller, 699
decaf::util::concurrent::atomic::AtomicReference, 590	activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller, 711
~BackupTransport	activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller, 695
activemq::transport::failover::BackupTransport, 592	activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller, 703
~BackupTransportPool	activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller, 707
activemq::transport::failover::BackupTransportPool, 595	~BrokerInfo
~BaseCommand	activemq::commands::BrokerInfo, 715

- ~BrokerInfoMarshaller
 - activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller, 727
 - activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller, 739
 - activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller, 723
 - activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller, 731
 - activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller, 735
- ~Buffer
 - decaf::nio::Buffer, 744
- ~BufferFactory
 - decaf::internal::nio::BufferFactory, 764
- ~BufferOverflowException
 - decaf::nio::BufferOverflowException, 773
- ~BufferUnderflowException
 - decaf::nio::BufferUnderflowException, 775
- ~BufferedInputStream
 - decaf::io::BufferedInputStream, 749
- ~BufferedOutputStream
 - decaf::io::BufferedOutputStream, 753
- ~BufferedSocket
 - decaf::net::BufferedSocket, 756
- ~Byte
 - decaf::lang::Byte, 778
- ~ByteArrayAdapter
 - decaf::internal::util::ByteArrayAdapter, 791
- ~ByteArrayBuffer
 - decaf::internal::nio::ByteArrayBuffer, 812
- ~ByteArrayInputStream
 - decaf::io::ByteArrayInputStream, 830
- ~ByteArrayOutputStream
 - decaf::io::ByteArrayOutputStream, 838
- ~ByteArrayPerspective
 - decaf::internal::nio::ByteArrayPerspective, 847
- ~ByteBuffer
 - decaf::nio::ByteBuffer, 853
- ~BytesMessage
 - cms::BytesMessage, 877
- ~CMSException
 - cms::CMSException, 962
- ~CMSExceptionSupport
 - activemq::util::CMSExceptionSupport, 964
- ~CMSProperties
 - cms::CMSProperties, 966
- ~CMSSecurityException
 - cms::CMSSecurityException, 969
- ~CachedConsumer
 - activemq::cmsutil::CachedConsumer, 889
- ~CachedProducer
 - activemq::cmsutil::CachedProducer, 892
- ~Callable
 - decaf::util::concurrent::Callable, 898
- ~CancellationException
 - decaf::util::concurrent::CancellationException, 900
- ~Certificate
 - decaf::security::cert::Certificate, 902
- ~CertificateEncodingException
 - decaf::security::cert::CertificateEncodingException, 906
- ~CertificateException
 - decaf::security::cert::CertificateException, 908
- ~CertificateExpiredException
 - decaf::security::cert::CertificateExpiredException, 910
- ~CertificateNotYetValidException
 - decaf::security::cert::CertificateNotYetValidException, 912
- ~CertificateParsingException
 - decaf::security::cert::CertificateParsingException, 914
- ~CharArrayBuffer
 - decaf::internal::nio::CharArrayBuffer, 925
- ~CharBuffer
 - decaf::nio::CharBuffer, 934
- ~CharSequence
 - decaf::lang::CharSequence, 947
- ~ClassCastException
 - decaf::lang::exceptions::ClassCastException, 950
- ~CloseTransportsTask
 - activemq::transport::failover::CloseTransportsTask, 953
- ~Closeable
 - cms::Closeable, 952
 - decaf::io::Closeable, 951
- ~CmsAccessor
 - activemq::cmsutil::CmsAccessor, 955
- ~CmsDestinationAccessor
 - activemq::cmsutil::CmsDestinationAccessor, 959
- ~CmsTemplate
 - activemq::cmsutil::CmsTemplate, 973
- ~Collection
 - decaf::util::Collection, 984
- ~Command
 - activemq::commands::Command, 992
- ~CommandVisitor
 - activemq::state::CommandVisitor, 998
- ~CommandVisitorAdapter
 - activemq::state::CommandVisitorAdapter, 1007

- ~Comparable
 - decaf::lang::Comparable, 1010
- ~Comparator
 - decaf::util::Comparator, 1012
- ~CompositeData
 - activemq::util::CompositeData, 1015
- ~CompositeTask
 - activemq::threads::CompositeTask, 1016
- ~CompositeTaskRunner
 - activemq::threads::CompositeTaskRunner, 1018
- ~CompositeTransport
 - activemq::transport::CompositeTransport, 1020
- ~ConcurrentMap
 - decaf::util::concurrent::ConcurrentMap, 1021
- ~ConcurrentStlMap
 - decaf::util::concurrent::ConcurrentStlMap, 1029
- ~Condition
 - decaf::util::concurrent::locks::Condition, 1042
- ~ConditionHandle
 - decaf::util::concurrent::ConditionHandle, 1047
- ~ConnectException
 - decaf::net::ConnectException, 1051
- ~Connection
 - cms::Connection, 1053
- ~ConnectionControl
 - activemq::commands::ConnectionControl, 1057
- ~ConnectionControlMarshaller
 - activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller, 1065
 - activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller, 1077
 - activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller, 1061
 - activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller, 1069
 - activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller, 1073
- ~ConnectionError
 - activemq::commands::ConnectionError, 1081
- ~ConnectionErrorMarshaller
 - activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller, 1088
 - activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller, 1100
 - activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller, 1084
- activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller, 1092
- activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller, 1096
- ~ConnectionFactory
 - cms::ConnectionFactory, 1104
- ~ConnectionId
 - activemq::commands::ConnectionId, 1107
- ~ConnectionIdMarshaller
 - activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller, 1114
 - activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller, 1126
 - activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller, 1110
 - activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller, 1118
 - activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller, 1122
- ~ConnectionInfo
 - activemq::commands::ConnectionInfo, 1130
- ~ConnectionInfoMarshaller
 - activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller, 1144
 - activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller, 1152
 - activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller, 1136
 - activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller, 1140
 - activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller, 1148
- ~ConnectionMetaData
 - activemq::commands::ConnectionMetaData, 1155
- ~ConnectionState
 - activemq::transport::ConnectionState, 1159
- ~ConnectionStateTracker
 - activemq::transport::ConnectionStateTracker, 1162
- ~ConsumerControl
 - activemq::commands::ConsumerControl, 1167
- ~ConsumerControlMarshaller
 - activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller, 1180
 - activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller, 1188
 - activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller, 1172
 - activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller, 1176
 - activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller, 1184

- ~ConsumerId
 - activemq::commands::ConsumerId, 1192
- ~ConsumerIdMarshaller
 - activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller, 1204
 - activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller, 1212
 - activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller, 1196
 - activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller, 1200
 - activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller, 1208
- ~ConsumerInfo
 - activemq::commands::ConsumerInfo, 1217
- ~ConsumerInfoMarshaller
 - activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller, 1228
 - activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller, 1240
 - activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller, 1224
 - activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller, 1232
 - activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller, 1236
- ~ConsumerState
 - activemq::state::ConsumerState, 1243
- ~ControlCommand
 - activemq::commands::ControlCommand, 1244
- ~ControlCommandMarshaller
 - activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller, 1255
 - activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller, 1263
 - activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller, 1247
 - activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller, 1251
 - activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller, 1259
- ~CountDownLatch
 - decaf::util::concurrent::CountDownLatch, 1266
- ~DataArrayResponse
 - activemq::commands::DataArrayResponse, 1269
- ~DataArrayResponseMarshaller
 - activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller, 1280
 - activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller, 1288
- activemq::wireformat::openwire::marshal::v3::DataArrayResponse, 1272
- activemq::wireformat::openwire::marshal::v4::DataArrayResponse, 1276
- activemq::wireformat::openwire::marshal::v5::DataArrayResponse, 1284
- ~DataInputStream
 - decaf::io::DataInputStream, 1293
- ~DataOutputStream
 - decaf::io::DataOutputStream, 1302
- ~DataResponse
 - activemq::commands::DataResponse, 1308
- ~DataResponseMarshaller
 - activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller, 1315
 - activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller, 1327
 - activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller, 1311
 - activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller, 1323
 - activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller, 1319
- ~DataStreamMarshaller
 - activemq::wireformat::openwire::marshal::DataStreamMarshaller, 1331
- ~DataStructure
 - activemq::commands::DataStructure, 1373
- ~Date
 - decaf::util::Date, 1379
- ~DecafRuntime
 - decaf::internal::DecafRuntime, 1382
- ~DedicatedTaskRunner
 - activemq::threads::DedicatedTaskRunner, 1383
- ~DefaultTransportListener
 - activemq::transport::DefaultTransportListener, 1384
- ~Delayed
 - decaf::util::concurrent::Delayed, 1386
- ~DeliveryMode
 - cms::DeliveryMode, 1387
- ~Destination
 - cms::Destination, 1389
- ~DestinationInfo
 - activemq::commands::DestinationInfo, 1392
- ~DestinationInfoMarshaller
 - activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller, 1408
 - activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller, 1396
 - activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller, 1400

- activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller, 1404
- activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller, 1412
- ~DestinationResolver
 - activemq::cmsutil::DestinationResolver, 1415
- ~DiscoveryEvent
 - activemq::commands::DiscoveryEvent, 1418
- ~DiscoveryEventMarshaller
 - activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller, 1437
 - activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller, 1421
 - activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller, 1425
 - activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller, 1429
 - activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller, 1433
- ~Dispatcher
 - activemq::core::Dispatcher, 1441
- ~Double
 - decaf::lang::Double, 1443
- ~DoubleArrayBuffer
 - decaf::internal::nio::DoubleArrayBuffer, 1455
- ~DoubleBuffer
 - decaf::nio::DoubleBuffer, 1462
- ~DynamicDestinationResolver
 - activemq::cmsutil::DynamicDestinationResolver, 1472
- ~EOFException
 - decaf::io::EOFException, 1475
- ~Entry
 - decaf::util::Map::Entry, 1473
- ~Exception
 - decaf::lang::Exception, 1479
- ~ExceptionListener
 - cms::ExceptionListener, 1483
- ~ExceptionResponse
 - activemq::commands::ExceptionResponse, 1485
- ~ExceptionResponseMarshaller
 - activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller, 1492
 - activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller, 1488
 - activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller, 1496
 - activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller, 1500
- activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller, 1504
- activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller, 1508
- ~Executor
 - decaf::util::concurrent::Executor, 1510
- ~ExecutorService
 - decaf::util::concurrent::ExecutorService, 1512
- ~FailoverTransport
 - activemq::wireformat::openwire::marshal::v1::FailoverTransportMarshaller, 1515
 - activemq::transport::failover::FailoverTransportFactory, 1524
 - ~FailoverTransportListener
 - activemq::wireformat::openwire::marshal::v1::FailoverTransportListenerMarshaller, 1526
 - ~Filter
 - decaf::util::logging::Filter, 1527
 - ~FilterInputStream
 - decaf::io::FilterInputStream, 1530
 - ~FilterOutputStream
 - decaf::io::FilterOutputStream, 1538
 - ~Float
 - decaf::lang::Float, 1546
 - ~FloatArrayBuffer
 - decaf::internal::nio::FloatArrayBuffer, 1557
 - ~FloatBuffer
 - decaf::nio::FloatBuffer, 1564
 - ~FlushCommand
 - activemq::commands::FlushCommand, 1574
 - ~FlushCommandMarshaller
 - activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller, 1581
 - activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller, 1577
 - activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller, 1585
 - activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller, 1589
 - activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller, 1593
 - ~Future
 - decaf::util::logging::Formatter, 1596
 - ~FutureResponse
 - decaf::util::concurrent::Future, 1598
 - ~FutureResponseMarshaller
 - activemq::transport::correlator::FutureResponseMarshaller, 1601
 - ~GeneralSecurityException

- decaf::security::GeneralSecurityException, 1604
- ~Handler
 - decaf::util::logging::Handler, 1605
- ~HexStringParser
 - decaf::internal::util::HexStringParser, 1608
- ~HexTable
 - activemq::wireformat::openwire::utils::HexTable, 1610
- ~HttpRequestException
 - decaf::net::HttpRequestException, 1612
- ~IOException
 - decaf::io::IOException, 1708
- ~IOTransport
 - activemq::transport::IOTransport, 1711
- ~IllegalArgumentException
 - decaf::lang::exceptions::IllegalArgumentException, 1615
- ~IllegalMonitorStateException
 - decaf::lang::exceptions::IllegalMonitorStateException, 1617
- ~IllegalStateException
 - cms::IllegalStateException, 1621
 - decaf::lang::exceptions::IllegalStateException, 1620
- ~IllegalThreadStateException
 - decaf::lang::exceptions::IllegalThreadStateException, 1623
- ~InactivityMonitor
 - activemq::transport::inactivity::InactivityMonitor, 1625
- ~IndexOutOfBoundsException
 - decaf::lang::exceptions::IndexOutOfBoundsException, 1629
- ~InputStream
 - decaf::io::InputStream, 1631
- ~IntArrayBuffer
 - decaf::internal::nio::IntArrayBuffer, 1637
- ~IntBuffer
 - decaf::nio::IntBuffer, 1644
- ~Integer
 - decaf::lang::Integer, 1655
- ~IntegerResponse
 - activemq::commands::IntegerResponse, 1668
- ~IntegerResponseMarshaller
 - activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller, 1675
 - activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller, 1671
 - activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller, 1679
 - activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller, 1683
- activemq::wireformat::openwire::marshal::v5::IntegerResponse, 1687
- ~InternalCommandListener
 - activemq::transport::mock::InternalCommandListener, 1690
- ~InterruptedException
 - decaf::lang::exceptions::InterruptedException, 1693
- ~InterruptedIOException
 - decaf::io::InterruptedIOException, 1695
- ~InvalidClientIdException
 - cms::InvalidClientIdException, 1697
- ~InvalidDestinationException
 - cms::InvalidDestinationException, 1698
- ~InvalidKeyException
 - decaf::security::InvalidKeyException, 1700
- ~InvalidMarkException
 - decaf::nio::InvalidMarkException, 1702
- ~InvalidSelectorException
 - cms::InvalidSelectorException, 1704
- ~InvalidStateException
 - decaf::lang::exceptions::InvalidStateException, 1706
- ~Iterable
 - decaf::lang::Iterable, 1716
- ~Iterator
 - decaf::util::Iterator, 1717
- ~JournalQueueAck
 - activemq::commands::JournalQueueAck, 1719
- ~JournalQueueAckMarshaller
 - activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller, 1735
 - activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller, 1723
 - activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller, 1731
 - activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller, 1727
 - activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller, 1739
- ~JournalTopicAck
 - activemq::commands::JournalTopicAck, 1743
- ~JournalTopicAckMarshaller
 - activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller, 1752
 - activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller, 1748
 - activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller, 1756
 - activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller, 1760

- activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller, 1764
- ~JournalTrace
 - activemq::commands::JournalTrace, 1767
- ~JournalTraceMarshaller
 - activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller, 1783
 - activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller, 1771
 - activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller, 1779
 - activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller, 1775
 - activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller, 1787
- ~JournalTransaction
 - activemq::commands::JournalTransaction, 1791
- ~JournalTransactionMarshaller
 - activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller, 1806
 - activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller, 1794
 - activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller, 1798
 - activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller, 1802
 - activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller, 1810
- ~KeepAliveInfo
 - activemq::commands::KeepAliveInfo, 1813
- ~KeepAliveInfoMarshaller
 - activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller, 1833
 - activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller, 1817
 - activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller, 1825
 - activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller, 1829
 - activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller, 1821
- ~Key
 - decaf::security::Key, 1837
- ~KeyException
 - decaf::security::KeyException, 1839
- ~LastPartialCommand
 - activemq::commands::LastPartialCommand, 1841
- ~LastPartialCommandMarshaller
 - activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller, 1848
 - activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller, 1844
- activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller, 1852
- activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller, 1856
- activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller, 1860
- ~Less
 - decaf::util::marshal::Less, 1863
- ~List
 - decaf::util::marshal::List, 1867
- ~ListIterator
 - decaf::util::marshal::ListIterator, 1872
- ~LocalTransactionId
 - activemq::commands::LocalTransactionId, 1876
- ~LocalTransactionIdMarshaller
 - activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller, 1891
 - activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller, 1879
 - activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller, 1883
 - activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller, 1885
 - activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller, 1895
- ~Lock
 - decaf::util::concurrent::locks::Lock, 1904
- ~LockSupport
 - decaf::util::concurrent::locks::LockSupport, 1906
- ~LogManager
 - decaf::util::logging::LogManager, 1925
- ~LogRecord
 - decaf::util::logging::LogRecord, 1929
- ~LogWriter
 - decaf::util::logging::LogWriter, 1933
- ~Logger
 - decaf::util::logging::Logger, 1910
- ~LoggerHierarchy
 - decaf::util::logging::LoggerHierarchy, 1917
- ~LoggingInputStream
 - activemq::io::LoggingInputStream, 1917
- ~LoggingOutputStream
 - activemq::io::LoggingOutputStream, 1919
- ~LoggingTransport
 - activemq::transport::logging::LoggingTransport, 1921
- ~Long
 - decaf::util::marshal::Long, 1937
- ~LongArrayBuffer
 - decaf::util::marshal::LongArrayBuffer, 1950
- ~LongBuffer
 - decaf::util::marshal::LongBuffer, 1950

- decaf::nio::LongBuffer, 1958
- ~LongSequenceGenerator
 - activemq::util::LongSequenceGenerator, 1967
- ~MalformedURLException
 - decaf::net::MalformedURLException, 1969
- ~Map
 - decaf::util::Map, 1972
- ~MapMessage
 - cms::MapMessage, 1985
- ~MarkBlockLogger
 - decaf::util::logging::MarkBlockLogger, 1992
- ~MarshalAware
 - activemq::wireformat::MarshalAware, 1993
- ~MarshallerFactory
 - activemq::wireformat::openwire::marshal::v1::MarshallerFactory, 1997
 - activemq::wireformat::openwire::marshal::v2::MarshallerFactory, 1996
 - activemq::wireformat::openwire::marshal::v3::MarshallerFactory, 1998
 - activemq::wireformat::openwire::marshal::v4::MarshallerFactory, 1996
 - activemq::wireformat::openwire::marshal::v5::MarshallerFactory, 1999
- ~Math
 - decaf::lang::Math, 2001
- ~MemoryUsage
 - activemq::util::MemoryUsage, 2016
- ~Message
 - activemq::commands::Message, 2022
 - cms::Message, 2040
- ~MessageAck
 - activemq::commands::MessageAck, 2056
- ~MessageAckMarshaller
 - activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller, 2077
 - activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller, 2061
 - activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller, 2069
 - activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller, 2073
 - activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller, 2065
- ~MessageConsumer
 - cms::MessageConsumer, 2081
- ~MessageCreator
 - activemq::cmsutil::MessageCreator, 2083
- ~MessageDispatch
 - activemq::commands::MessageDispatch, 2085
- ~MessageDispatchChannel
 - activemq::core::MessageDispatchChannel, 2090
- ~MessageDispatchMarshaller
 - activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller, 2109
 - activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller, 2097
 - activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller, 2105
 - activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller, 2101
 - activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller, 2113
- ~MessageDispatchNotification
 - activemq::commands::MessageDispatchNotification, 2117
- ~MessageDispatchNotificationMarshaller
 - activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller, 2133
 - activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller, 2121
 - activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller, 2129
 - activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller, 2123
 - activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller, 2137
- ~MessageEOFException
 - cms::MessageEOFException, 2141
- ~MessageFormatException
 - cms::MessageFormatException, 2142
- ~MessageId
 - activemq::commands::MessageId, 2143
- ~MessageIdMarshaller
 - activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller, 2163
 - activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller, 2147
 - activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller, 2155
 - activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller, 2151
 - activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller, 2159
- ~MessageListener
 - cms::MessageListener, 2166
- ~MessageMarshaller
 - activemq::wireformat::openwire::marshal::v1::MessageMarshaller, 2184
 - activemq::wireformat::openwire::marshal::v2::MessageMarshaller, 2167
 - activemq::wireformat::openwire::marshal::v3::MessageMarshaller, 2176

- activemq::wireformat::openwire::marshal::v4::MessageMarshaller, NoSuchAlgorithmException, 2172
- activemq::wireformat::openwire::marshal::v5::MessageMarshaller, NoSuchAlgorithmException, 2180
- ~MessageNotReadableException
 - cms::MessageNotReadableException, 2188
- ~MessageNotWriteableException
 - cms::MessageNotWriteableException, 2189
- ~MessageProducer
 - cms::MessageProducer, 2191
- ~MessagePropertyInterceptor
 - activemq::wireformat::openwire::utils::MessagePropertyInterceptor, 2198
- ~MessagePull
 - activemq::commands::MessagePull, 2204
- ~MessagePullMarshaller
 - activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller, 2216
 - activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller, 2208
 - activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller; openssl::OpenSSLX500Principal, 2212
 - activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller; openssl::OpenSSLX509Certificate, 2224
 - activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller; openssl::OpenSSLX509Certificate, 2220
- ~MockTransport
 - activemq::transport::mock::MockTransport, 2229
- ~MockTransportFactory
 - activemq::transport::mock::MockTransportFactory, 2238
- ~Mutex
 - decaf::util::concurrent::Mutex, 2240
- ~MutexHandle
 - decaf::util::concurrent::MutexHandle, 2244
- ~NetworkBridgeFilter
 - activemq::commands::NetworkBridgeFilter, 2247
- ~NetworkBridgeFilterMarshaller
 - activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller, 2263
 - activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller, 2251
 - activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller, 2255
 - activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller, 2267
 - activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller, 2259
- ~NoRouteToHostException
 - decaf::net::NoRouteToHostException, 2271
- ~NoSuchAlgorithmException
 - decaf::lang::exceptions::NoSuchAlgorithmException, 2274
 - decaf::lang::exceptions::NoSuchElementException, 2276
 - ~NoSuchProviderException
 - decaf::security::NoSuchProviderException, 2279
 - ~NullPointerException
 - decaf::lang::exceptions::NullPointerException, 2281
 - ~NumberFormatException
 - decaf::lang::exceptions::NumberFormatException, 2286
 - ~ObjectMessage
 - cms::ObjectMessage, 2287
 - ~OpenSSLX500Principal
 - decaf::security_ - 2288
 - ~OpenSSLX509Certificate
 - decaf::security_ - 2291
 - ~OpenWireFormat
 - activemq::wireformat::openwire::OpenWireFormat, 2299
 - ~OpenWireFormatFactory
 - activemq::wireformat::openwire::OpenWireFormatFactory, 2309
 - ~OpenWireFormatNegotiator
 - activemq::wireformat::openwire::OpenWireFormatNegotiator, 2311
 - ~OpenWireResponseBuilder
 - activemq::wireformat::openwire::OpenWireResponseBuilder, 2314
 - ~OpenwireStringSupport
 - activemq::wireformat::openwire::utils::OpenwireStringSupport, 2315
 - ~OutputStream
 - decaf::io::OutputStream, 2317
 - ~PartialCommand
 - activemq::wireformat::openwire::marshal::v1::PartialCommand, 2326
 - activemq::wireformat::openwire::marshal::v2::PartialCommand, 2323
 - activemq::wireformat::openwire::marshal::v3::PartialCommand, 2327

- activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller, 2463
- activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller, 2451
- ~Pointer
 - decaf::lang::Pointer, 2347
- ~PooledSession
 - activemq::cmsutil::PooledSession, 2354
- ~PooledThread
 - decaf::util::concurrent::PooledThread, 2364
- ~PooledThreadListener
 - decaf::util::concurrent::PooledThreadListener, 2367
- ~PortUnreachableException
 - decaf::net::PortUnreachableException, 2369
- ~PrimitiveList
 - activemq::util::PrimitiveList, 2373
- ~PrimitiveMap
 - activemq::util::PrimitiveMap, 2383
- ~PrimitiveTypesMarshaller
 - activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller, 2392
- ~PrimitiveValueConverter
 - activemq::util::PrimitiveValueConverter, 2397
- ~PrimitiveValueNode
 - activemq::util::PrimitiveValueNode, 2404
- ~Principal
 - decaf::security::Principal, 2412
- ~PriorityQueue
 - decaf::util::PriorityQueue, 2416
- ~ProducerAck
 - activemq::commands::ProducerAck, 2421
- ~ProducerAckMarshaller
 - activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller, 2441
 - activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller, 2425
 - activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller, 2429
 - activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller, 2433
 - activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller, 2437
- ~ProducerCallback
 - activemq::cmsutil::ProducerCallback, 2444
- ~ProducerExecutor
 - activemq::cmsutil::CmsTemplate::ProducerExecutor, 2445
- ~ProducerId
 - activemq::commands::ProducerId, 2447
- ~ProducerIdMarshaller
 - activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller, 2463
 - activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller, 2455
 - activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller, 2467
 - activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller, 2459
 - ~ProducerInfo
 - activemq::commands::ProducerInfo, 2471
 - ~ProducerInfoMarshaller
 - activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller, 2491
 - activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller, 2475
 - activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller, 2479
 - activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller, 2487
 - activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller, 2483
 - ~ProducerState
 - activemq::state::ProducerState, 2494
 - ~Properties
 - decaf::util::Properties, 2496
 - ~PropertiesChangeListener
 - decaf::util::logging::PropertiesChangeListener, 2504
 - ~ProtocolException
 - decaf::net::ProtocolException, 2506
 - ~PublicKey
 - decaf::security::PublicKey, 2507
 - ~Queue
 - activemq::util::Queue, 2511
 - decaf::util::Queue, 2508
 - ~QueueBrowserMarshaller
 - cms::QueueBrowser, 2512
 - ~ReadCheckerMarshaller
 - activemq::transport::inactivity::ReadChecker, 2518
 - ~ReadOnlyBufferException
 - decaf::io::ReadOnlyBufferException, 2522
 - ~ReadWriteLock
 - decaf::util::concurrent::locks::ReadWriteLock, 2524
 - ~Reader
 - decaf::io::Reader, 2519
 - ~ReceiveExecutor
 - activemq::cmsutil::CmsTemplate::ReceiveExecutor, 2525
 - ~ReentrantLock

- decaf::util::concurrent::locks::ReentrantLock, 2528
- ~RejectedExecutionException
 - decaf::util::concurrent::RejectedExecutionException, 2535
- ~RejectedExecutionHandler
 - decaf::util::concurrent::RejectedExecutionHandler, 2536
- ~RemoveInfo
 - activemq::commands::RemoveInfo, 2538
- ~RemoveInfoMarshaller
 - activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller, 2541
 - activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller, 2549
 - activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller, 2553
 - activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller, 2557
 - activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller, 2545
- ~RemoveSubscriptionInfo
 - activemq::commands::RemoveSubscriptionInfo, 2561
- ~RemoveSubscriptionInfoMarshaller
 - activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller, 2573
 - activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller, 2569
 - activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller, 2577
 - activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller, 2581
 - activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller, 2565
- ~ReplayCommand
 - activemq::commands::ReplayCommand, 2585
- ~ReplayCommandMarshaller
 - activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller, 2604
 - activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller, 2588
 - activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller, 2592
 - activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller, 2600
 - activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller, 2596
- ~ResolveProducerExecutor
 - activemq::cmsutil::CmsTemplate::ResolveProducerExecutor, 2607
- ~ResolveReceiveExecutor
 - activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor, 2608
- ~ResourceLifecycleManager
 - activemq::cmsutil::ResourceLifecycleManager, 2609
- ~Response
 - activemq::commands::Response, 2612
- ~ResponseBuilder
 - activemq::transport::mock::ResponseBuilder, 2615
- ~ResponseCorrelator
 - activemq::transport::correlator::ResponseCorrelator, 2617
- ~ResponseMarshaller
 - activemq::wireformat::openwire::marshal::v1::ResponseMarshaller, 2634
 - activemq::wireformat::openwire::marshal::v2::ResponseMarshaller, 2621
 - activemq::wireformat::openwire::marshal::v3::ResponseMarshaller, 2625
 - activemq::wireformat::openwire::marshal::v4::ResponseMarshaller, 2639
 - activemq::wireformat::openwire::marshal::v5::ResponseMarshaller, 2630
- ~Runnable
 - decaf::lang::Runnable, 2642
- ~Runtime
 - decaf::lang::Runtime, 2643
- ~RuntimeException
 - decaf::lang::exceptions::RuntimeException, 2646
- ~SecurityProvider
 - decaf::security_provider::SecurityProvider, 2647
- ~SecurityProviderRegistrar
 - decaf::security_provider::SecurityProviderRegistrar, 2650
- ~Semaphore
 - decaf::util::concurrent::Semaphore, 2654
- ~SendCommand
 - activemq::cmsutil::CmsTemplate::SendExecutor, 2661
- ~ServerSocket
 - decaf::net::ServerSocket, 2662
- ~Session
 - activemq::commands::Session, 2667
- ~SessionCallback
 - activemq::commands::SessionCallback, 2677
- ~SessionId
 - activemq::commands::SessionId, 2679
- ~SessionIdMarshaller
 - activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller, 2687

- activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller, 2787
- 2691 ~SocketException
- activemq::wireformat::openwire::marshal::v3::SessionInfoMarshallerException, 2794
- 2699 ~SocketFactory
- activemq::wireformat::openwire::marshal::v4::SessionInfoMarshallerFactory, 2796
- 2695 ~SocketInputStream
- activemq::wireformat::openwire::marshal::v5::SessionInfoMarshallerInputStream, 2798
- 2683 ~SocketOutputStream
- ~SessionInfo
 - activemq::commands::SessionInfo, 2703
- ~SessionInfoMarshaller
 - activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller, 2710
 - activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller, 2706
 - activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller, 2722
 - activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller, 2714
 - activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller, 2718
- ~SessionPool
 - activemq::cmsutil::SessionPool, 2726
- ~SessionState
 - activemq::state::SessionState, 2728
- ~Set
 - decaf::util::Set, 2729
- ~Short
 - decaf::lang::Short, 2732
- ~ShortArrayBuffer
 - decaf::internal::nio::ShortArrayBuffer, 2741
- ~ShortBuffer
 - decaf::nio::ShortBuffer, 2748
- ~ShutdownInfo
 - activemq::commands::ShutdownInfo, 2758
- ~ShutdownInfoMarshaller
 - activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller, 2761
 - activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller, 2773
 - activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller, 2769
 - activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller, 2765
 - activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller, 2777
- ~SignatureException
 - decaf::security::SignatureException, 2781
- ~SimpleFormatter
 - decaf::util::logging::SimpleFormatter, 2783
- ~SimpleLogger
 - decaf::util::logging::SimpleLogger, 2785
- ~Socket
 - decaf::net::Socket, 2805
 - decaf::net::SocketException, 2811
 - decaf::internal::io::StandardErrorOutputStream, 2814
 - decaf::io::StandardInputStream, 2820
 - decaf::io::StandardOutputStream, 2826
 - ~Startable
 - cms::Startable, 2831
 - ~StaticInitializer
 - activemq::core::ActiveMQConstants::StaticInitializer, 2832
 - ~StlList
 - decaf::util::StlList, 2838
 - ~StlMap
 - decaf::util::StlMap, 2849
 - ~StlQueue
 - decaf::util::StlQueue, 2860
 - ~StlSet
 - decaf::util::StlSet, 2868
 - ~StompFrame
 - activemq::wireformat::stomp::StompFrame, 2875
 - ~StompHelper
 - activemq::wireformat::stomp::StompHelper, 2880
 - ~StompWireFormat
 - activemq::wireformat::stomp::StompWireFormat, 2886
 - ~StompWireFormatFactory
 - activemq::wireformat::stomp::StompWireFormatFactory, 2887
 - ~Stoppable
 - cms::Stoppable, 2888
 - ~StreamHandler
 - decaf::util::logging::StreamHandler, 2889
 - ~StreamMessage
 - cms::StreamMessage, 2895
 - ~StringTokenizer
 - decaf::util::StringTokenizer, 2905
 - ~SubscriptionInfo

- activemq::commands::SubscriptionInfo, 2908
- ~SubscriptionInfoMarshaller
 - activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller, 2912
 - activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller, 2928
 - activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller, 2916
 - activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller, 2924
 - activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller, 2920
- ~Synchronizable
 - decaf::util::concurrent::Synchronizable, 2932
- ~SynchronizableImpl
 - decaf::internal::util::concurrent::SynchronizableImpl, 2943
- ~Synchronization
 - activemq::core::Synchronization, 2946
- ~SynchronousQueue
 - decaf::util::concurrent::SynchronousQueue, 2949
- ~System
 - decaf::lang::System, 2954
- ~Task
 - activemq::threads::Task, 2956
- ~TaskListener
 - decaf::util::concurrent::TaskListener, 2957
- ~TaskRunner
 - activemq::threads::TaskRunner, 2958
- ~TcpSocket
 - decaf::net::TcpSocket, 2961
- ~TcpTransport
 - activemq::transport::tcp::TcpTransport, 2968
- ~TcpTransportFactory
 - activemq::transport::tcp::TcpTransportFactory, 2970
- ~TemporaryQueue
 - cms::TemporaryQueue, 2972
- ~TemporaryTopic
 - cms::TemporaryTopic, 2973
- ~TextMessage
 - cms::TextMessage, 2975
- ~ThreadFactory
 - decaf::util::concurrent::ThreadFactory, 2976
- ~ThreadGroup
 - decaf::lang::ThreadGroup, 2977
- ~ThreadPool
 - decaf::util::concurrent::ThreadPool, 2979
- ~Throwable
 - decaf::lang::Throwable, 2984
- ~TimeUnit
 - decaf::util::concurrent::TimeUnit, 3007
 - decaf::util::concurrent::TimeoutException, 2989
- ~Timer
 - decaf::util::Timer, 2992
- ~TimerTask
 - decaf::util::TimerTask, 3001
- ~TimerTaskHeap
 - decaf::util::TimerTaskHeap, 3003
- ~Topic
 - cms::Topic, 3014
- ~Tracked
 - activemq::state::Tracked, 3015
- ~TransactionId
 - activemq::commands::TransactionId, 3017
- ~TransactionIdMarshaller
 - activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller, 3023
 - activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller, 3020
 - activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller, 3034
 - activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller, 3027
 - activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller, 3031
- ~TransactionInfo
 - activemq::commands::TransactionInfo, 3038
- ~TransactionInfoMarshaller
 - activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller, 3050
 - activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller, 3058
 - activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller, 3046
 - activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller, 3054
 - activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller, 3042
- ~TransactionState
 - activemq::state::TransactionState, 3061
- ~TransferQueue
 - decaf::internal::util::concurrent::TransferQueue, 3063
- ~TransferStack
 - decaf::internal::util::concurrent::TransferStack, 3065
- ~Transport
 - activemq::transport::Transport, 3067
- ~TransportFactory

- activemq::transport::TransportFactory, 3072
- ~TransportFilter
 - activemq::transport::TransportFilter, 3075
- ~TransportListener
 - activemq::transport::TransportListener, 3081
- ~TransportRegistry
 - activemq::transport::TransportRegistry, 3083
- ~URI
 - decaf::net::URI, 3100
- ~URIEncoderDecoder
 - decaf::internal::net::URIEncoderDecoder, 3108
- ~URIHelper
 - decaf::internal::net::URIHelper, 3112
- ~URIPool
 - activemq::transport::failover::URIPool, 3118
- ~URISyntaxException
 - decaf::net::URISyntaxException, 3125
- ~URIType
 - decaf::internal::net::URIType, 3128
- ~URL
 - decaf::net::URL, 3135
- ~URLDecoder
 - decaf::net::URLDecoder, 3135
- ~URLEncoder
 - decaf::net::URLEncoder, 3136
- ~UTFDataFormatException
 - decaf::io::UTFDataFormatException, 3141
- ~UUID
 - decaf::util::UUID, 3143
- ~UnknownHostException
 - decaf::net::UnknownHostException, 3087
- ~UnknownServiceException
 - decaf::net::UnknownServiceException, 3089
- ~UnsupportedEncodingException
 - decaf::io::UnsupportedEncodingException, 3092
- ~UnsupportedOperationException
 - cms::UnsupportedOperationException, 3093
 - decaf::lang::exceptions::UnsupportedOperationException, 3096
- ~Usage
 - activemq::util::Usage, 3137
- ~WireFormat
 - activemq::wireformat::WireFormat, 3149
- ~WireFormatFactory
 - activemq::wireformat::WireFormatFactory, 3152
- ~WireFormatInfo
 - activemq::commands::WireFormatInfo, 3155
- ~WireFormatInfoMarshaller
 - activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller, 3172
 - activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller, 3164
 - activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller, 3180
 - activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller, 3176
 - activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller, 3168
- ~WireFormatNegotiator
 - activemq::wireformat::WireFormatNegotiator, 3183
- ~WireFormatRegistry
 - activemq::wireformat::WireFormatRegistry, 3184
- ~WriteChecker
 - activemq::transport::inactivity::WriteChecker, 3186
- ~Writer
 - decaf::io::Writer, 3187
- ~X500Principal
 - decaf::security::auth::x500::X500Principal, 3189
- ~X509Certificate
 - decaf::security::cert::X509Certificate, 3190
- ~XATransactionId
 - activemq::commands::XATransactionId, 3194
- ~XATransactionIdMarshaller
 - activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller, 3210
 - activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller, 3198
 - activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller, 3202
 - activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller, 3206
 - activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller, 3214
- _FALSE
 - decaf::lang::Boolean, 678
- _TRUE
 - decaf::lang::Boolean, 678
- _array
 - decaf::internal::nio::CharArrayBuffer, 930
- _capacity
 - decaf::nio::Buffer, 747
- _limit
 - decaf::nio::Buffer, 747

- `_mark`
 - `decaf::nio::Buffer`, 747
- `_markSet`
 - `decaf::nio::Buffer`, 747
- `_position`
 - `decaf::nio::Buffer`, 747
- ABORT
 - `activemq::wireformat::stomp::StompCommandConstants`, 2873
- abs
 - `decaf::lang::Math`, 2001, 2002
- AbstractQueue
 - `decaf::util::AbstractQueue`, 135
- accept
 - `decaf::net::ServerSocket`, 2662
- ACK
 - `activemq::wireformat::stomp::StompCommandConstants`, 2873
- ACK_AUTO
 - `activemq::wireformat::stomp::StompCommandConstants`, 2873
- ACK_CLIENT
 - `activemq::wireformat::stomp::StompCommandConstants`, 2873
- ACK_INDIVIDUAL
 - `activemq::wireformat::stomp::StompCommandConstants`, 2873
- ACK_TYPE_CONSUMED
 - `activemq::core::ActiveMQConstants`, 228
- ACK_TYPE_DELIVERED
 - `activemq::core::ActiveMQConstants`, 228
- ACK_TYPE_INDIVIDUAL
 - `activemq::core::ActiveMQConstants`, 228
- ACK_TYPE_POISON
 - `activemq::core::ActiveMQConstants`, 228
- ACK_TYPE_REDELIVERED
 - `activemq::core::ActiveMQConstants`, 228
- acknowledge
 - `activemq::commands::ActiveMQMessageTemplate`, 331
 - `activemq::core::ActiveMQConsumer`, 232, 233
 - `activemq::core::ActiveMQSession`, 406
 - `cms::Message`, 2040
- acknowledgeMessage
 - `activemq::core::ActiveMQAckHandler`, 141
- AcknowledgeMode
 - `cms::Session`, 2667
- AckType
 - `activemq::core::ActiveMQConstants`, 228
- ackType
 - `activemq::commands::MessageAck`, 2060
- acquire
 - `decaf::util::concurrent::Semaphore`, 2654
- acquireUninterruptibly
 - `decaf::util::concurrent::Semaphore`, 2655
- action
 - `activemq::cmsutil::CmsTemplate::ProducerExecutor`, 2446
- activemq, 69
- `activemq/exceptions/ExceptionDefines.h`
- `AMQ_CATCH_EXCEPTION_-CONVERT`, 3272
- `AMQ_CATCH_NOTHROW`, 3273
- `AMQ_CATCH_RETHROW`, 3273
- `AMQ_CATCHALL_NOTHROW`, 3273
- `AMQ_CATCHALL_THROW`, 3273
- `activemq/util/Config.h`
- `AMQCPP_API`, 3306
- `HAVE_PTHREAD_H`, 3306
- `HAVE_UUID_T`, 3306
- `HAVE_UUID_UUID_H`, 3306
- `activemq::cmsutil`, 70
- `activemq::cmsutil::CachedConsumer`, 888
- `~CachedConsumer`, 889
- `CachedConsumer`, 889
- `close`, 889
- `getMessageListener`, 889
- `getMessageSelector`, 889
- `retain`, 889, 890
- `receiveNoWait`, 890
- `setMessageListener`, 890
- `activemq::cmsutil::CachedProducer`, 891
- `~CachedProducer`, 892
- `CachedProducer`, 892
- `close`, 892
- `getDeliveryMode`, 892
- `getDisableMessageID`, 893
- `getDisableMessageTimeStamp`, 893
- `getPriority`, 893
- `getTimeToLive`, 893
- `send`, 894, 895
- `setDeliveryMode`, 896
- `setDisableMessageID`, 896
- `setDisableMessageTimeStamp`, 896
- `setPriority`, 896
- `setTimeToLive`, 897
- `activemq::cmsutil::CmsAccessor`, 954
- `~CmsAccessor`, 955
- `checkConnectionFactory`, 955
- `CmsAccessor`, 955
- `createConnection`, 955
- `createSession`, 955
- `destroy`, 956
- `getConnectionFactory`, 956
- `getResourceLifecycleManager`, 956
- `getSessionAcknowledgeMode`, 956

- init, 957
- setConnectionFactory, 957
- setSessionAcknowledgeMode, 957
- activemq::cmsutil::CmsDestinationAccessor, 957
 - ~CmsDestinationAccessor, 959
 - checkDestinationResolver, 959
 - CmsDestinationAccessor, 959
 - destroy, 959
 - getDestinationResolver, 959
 - init, 959
 - isPubSubDomain, 959
 - resolveDestinationName, 960
 - setDestinationResolver, 960
 - setPubSubDomain, 960
- activemq::cmsutil::CmsTemplate, 969
 - ~CmsTemplate, 973
 - CmsTemplate, 973
 - DEFAULT_PRIORITY, 981
 - DEFAULT_TIME_TO_LIVE, 981
 - destroy, 973
 - execute, 973, 974
 - getDefaultDestination, 974
 - getDefaultDestinationName, 975
 - getDeliveryMode, 975
 - getPriority, 975
 - getReceiveTimeout, 975
 - getTimeToLive, 975
 - init, 975
 - isExplicitQosEnabled, 975
 - isMessageIdEnabled, 976
 - isMessageTimestampEnabled, 976
 - isNoLocal, 976
 - ProducerExecutor, 981
 - receive, 976, 977
 - RECEIVE_TIMEOUT_INDEFINITE_WAIT, 981
 - RECEIVE_TIMEOUT_NO_WAIT, 982
 - ReceiveExecutor, 981
 - receiveSelected, 977, 978
 - ResolveProducerExecutor, 981
 - ResolveReceiveExecutor, 981
 - send, 978
 - SendExecutor, 981
 - setDefaultDestination, 979
 - setDefaultDestinationName, 979
 - setDeliveryMode, 979
 - setDeliveryPersistent, 979
 - setExplicitQosEnabled, 980
 - setMessageIdEnabled, 980
 - setMessageTimestampEnabled, 980
 - setNoLocal, 980
 - setPriority, 980
 - setPubSubDomain, 980
 - setReceiveTimeout, 981
 - setTimeToLive, 981
- activemq::cmsutil::CmsTemplate::ProducerExecutor, 2444
 - ~ProducerExecutor, 2445
 - action, 2446
 - destination, 2446
 - doInCms, 2445
 - getDestination, 2445
 - parent, 2446
 - ProducerExecutor, 2445
- activemq::cmsutil::CmsTemplate::ReceiveExecutor, 2524
 - ~ReceiveExecutor, 2525
 - destination, 2526
 - doInCms, 2525
 - getDestination, 2525
 - getMessage, 2526
 - message, 2526
 - noLocal, 2526
 - parent, 2526
 - ReceiveExecutor, 2525
 - selector, 2526
- activemq::cmsutil::CmsTemplate::ResolveProducerExecutor, 2607
 - ~ResolveProducerExecutor, 2607
 - getDestination, 2607
 - ResolveProducerExecutor, 2607
- activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor, 2608
 - ~ResolveReceiveExecutor, 2608
 - getDestination, 2608
 - ResolveReceiveExecutor, 2608
- activemq::cmsutil::CmsTemplate::SendExecutor, 2660
 - ~SendExecutor, 2661
 - doInCms, 2661
 - SendExecutor, 2661
- activemq::cmsutil::DestinationResolver, 1415
 - ~DestinationResolver, 1415
 - destroy, 1415
 - init, 1416
 - resolveDestinationName, 1416
- activemq::cmsutil::DynamicDestinationResolver, 1471
 - ~DynamicDestinationResolver, 1472
 - destroy, 1472
 - init, 1472
 - resolveDestinationName, 1472
- activemq::cmsutil::MessageCreator, 2083
 - ~MessageCreator, 2083
 - createMessage, 2083
- activemq::cmsutil::PooledSession, 2351
 - ~PooledSession, 2354

- close, 2354
- commit, 2354
- createBrowser, 2354, 2355
- createBytesMessage, 2355
- createCachedConsumer, 2356
- createCachedProducer, 2356
- createConsumer, 2356, 2357
- createDurableConsumer, 2358
- createMapMessage, 2358
- createMessage, 2359
- createProducer, 2359
- createQueue, 2359
- createStreamMessage, 2360
- createTemporaryQueue, 2360
- createTemporaryTopic, 2360
- createTextMessage, 2361
- createTopic, 2361
- getAcknowledgeMode, 2361
- getSession, 2362
- isTransacted, 2362
- PooledSession, 2354
- recover, 2362
- rollback, 2363
- unsubscribe, 2363
- activemq::cmsutil::ProducerCallback, 2443
 - ~ProducerCallback, 2444
 - doInCms, 2444
- activemq::cmsutil::ResourceLifecycleManager, 2608
 - ~ResourceLifecycleManager, 2609
 - addConnection, 2609
 - addDestination, 2609
 - addMessageConsumer, 2610
 - addMessageProducer, 2610
 - addSession, 2610
 - destroy, 2610
 - releaseAll, 2610
 - ResourceLifecycleManager, 2609
- activemq::cmsutil::SessionCallback, 2676
 - ~SessionCallback, 2677
 - doInCms, 2677
- activemq::cmsutil::SessionPool, 2725
 - ~SessionPool, 2726
 - getResourceLifecycleManager, 2726
 - returnSession, 2726
 - SessionPool, 2725
 - takeSession, 2726
- activemq::commands, 71
- activemq::commands::ActiveMQBlobMessage, 142
 - ~ActiveMQBlobMessage, 143
 - ActiveMQBlobMessage, 143
 - BINARY_MIME_TYPE, 146
 - clone, 143
 - cloneDataStructure, 143
 - copyDataStructure, 143
 - equals, 144
 - getDataStructureType, 144
 - getMimeType, 144
 - getName, 144
 - getRemoteBlobUrl, 144
 - ID_ACTIVEMQBLOBMESSAGE, 146
 - isDeletedByBroker, 145
 - setDeletedByBroker, 145
 - setMimeType, 145
 - setName, 145
 - setRemoteBlobUrl, 145
 - toString, 146
- activemq::commands::ActiveMQBytesMessage, 166
 - ~ActiveMQBytesMessage, 170
 - ActiveMQBytesMessage, 170
 - clearBody, 170
 - clone, 170
 - cloneDataStructure, 170
 - copyDataStructure, 170
 - equals, 171
 - getBodyBytes, 171
 - getBodyLength, 171
 - getDataStructureType, 172
 - ID_ACTIVEMQBYTESMESSAGE, 182
 - onSend, 172
 - readBoolean, 172
 - readByte, 172
 - readBytes, 173
 - readChar, 174
 - readDouble, 174
 - readFloat, 174
 - readInt, 175
 - readLong, 175
 - readShort, 175
 - readString, 176
 - readUnsignedShort, 176
 - readUTF, 176
 - reset, 177
 - setBodyBytes, 177
 - toString, 177
 - writeBoolean, 178
 - writeByte, 178
 - writeBytes, 178, 179
 - writeChar, 179
 - writeDouble, 179
 - writeFloat, 179
 - writeInt, 180
 - writeLong, 180
 - writeShort, 180
 - writeString, 181
 - writeUnsignedShort, 181

- writeUTF, 181
- activemq::commands::ActiveMQDestination, 240
 - ~ActiveMQDestination, 242
 - ActiveMQDestination, 242
 - advisory, 248
 - ADVISORY_PREFIX, 248
 - cloneDataStructure, 242
 - COMPOSITE_SEPARATOR, 248
 - CONNECTION_ADVISORY_PREFIX, 248
 - CONSUMER_ADVISORY_PREFIX, 248
 - copyDataStructure, 242
 - createDestination, 243
 - createTemporaryName, 243
 - DEFAULT_ORDERED_TARGET, 249
 - equals, 243
 - exclusive, 249
 - getClientId, 244
 - getCMSDestination, 244
 - getDataStructureType, 244
 - getDestinationType, 244
 - getOptions, 244
 - getOrderedTarget, 245
 - getPhysicalName, 245
 - ID_ACTIVEMQDESTINATION, 249
 - isAdvisory, 245
 - isComposite, 245
 - isConnectionAdvisory, 245
 - isConsumerAdvisory, 246
 - isExclusive, 246
 - isOrdered, 246
 - isProducerAdvisory, 246
 - isQueue, 246
 - isTemporary, 246
 - isTopic, 247
 - isWildcard, 247
 - options, 249
 - ordered, 249
 - orderedTarget, 249
 - physicalName, 249
 - PRODUCER_ADVISORY_PREFIX, 249
 - QUEUE_QUALIFIED_PREFIX, 249
 - setAdvisory, 247
 - setExclusive, 247
 - setOrdered, 247
 - setOrderedTarget, 247
 - setPhysicalName, 248
 - TEMP_POSTFIX, 250
 - TEMP_PREFIX, 250
 - TEMP_QUEUE_QUALIFIED_PREFIX, 250
 - TEMP_TOPIC_QUALIFIED_PREFIX, 250
 - TOPIC_QUALIFIED_PREFIX, 250
 - toString, 248
- activemq::commands::ActiveMQDestination::DestinationFilter, 1390
 - ANY_CHILD, 1390
 - ANY_DESCENDENT, 1390
- activemq::commands::ActiveMQMapMessage, 272
 - ~ActiveMQMapMessage, 275
 - ActiveMQMapMessage, 275
 - beforeMarshal, 275
 - checkMapIsUnmarshalled, 275
 - clearBody, 275
 - clone, 275
 - cloneDataStructure, 275
 - copyDataStructure, 276
 - equals, 276
 - getBoolean, 276
 - getByte, 276
 - getBytes, 277
 - getChar, 277
 - getDataStructureType, 277
 - getDouble, 278
 - getFloat, 278
 - getInt, 278
 - getLong, 279
 - getMap, 279
 - getMapNames, 279
 - getShort, 279
 - getString, 280
 - ID_ACTIVEMQMAPMESSAGE, 284
 - isMarshalAware, 280
 - itemExists, 280
 - setBoolean, 281
 - setByte, 281
 - setBytes, 281
 - setChar, 282
 - setDouble, 282
 - setFloat, 282
 - setInt, 283
 - setLong, 283
 - setShort, 283
 - setString, 283
 - toString, 284
- activemq::commands::ActiveMQMessage, 304
 - ~ActiveMQMessage, 305
 - ActiveMQMessage, 305
 - clone, 305
 - cloneDataStructure, 305
 - copyDataStructure, 306
 - equals, 306
 - getDataStructureType, 306
 - ID_ACTIVEMQMESSAGE, 307
 - toString, 306

- activemq::commands::ActiveMQMessageTemplate, 327
 - ~ActiveMQMessageTemplate, 331
 - acknowledge, 331
 - ActiveMQMessageTemplate, 331
 - clearBody, 331
 - clearProperties, 331
 - equals, 331
 - failIfReadOnlyBody, 332
 - failIfReadOnlyProperties, 332
 - failIfWriteOnlyBody, 332
 - getBooleanProperty, 332
 - getByteProperty, 332
 - getCMSCorrelationID, 333
 - getCMSDeliveryMode, 333
 - getCMSDestination, 333
 - getCMSExpiration, 333
 - getCMSMessageID, 334
 - getCMSPriority, 334
 - getCMSRedelivered, 334
 - getCMSReplyTo, 335
 - getCMSTimestamp, 335
 - getCMSType, 335
 - getDoubleProperty, 335
 - getFloatProperty, 336
 - getIntProperty, 336
 - getLongProperty, 337
 - getPropertyNames, 337
 - getShortProperty, 337
 - getStringProperty, 337
 - onSend, 338
 - propertyExists, 338
 - setBooleanProperty, 338
 - setByteProperty, 339
 - setCMSCorrelationID, 339
 - setCMSDeliveryMode, 339
 - setCMSDestination, 340
 - setCMSExpiration, 340
 - setCMSMessageID, 340
 - setCMSPriority, 340
 - setCMSRedelivered, 341
 - setCMSReplyTo, 341
 - setCMSTimestamp, 341
 - setCMSType, 342
 - setDoubleProperty, 342
 - setFloatProperty, 342
 - setIntProperty, 343
 - setLongProperty, 343
 - setShortProperty, 343
 - setStringProperty, 344
- activemq::commands::ActiveMQObjectMessage, 344
 - ~ActiveMQObjectMessage, 345
 - ActiveMQObjectMessage, 345
 - clone, 345
 - cloneDataStructure, 345
 - copyDataStructure, 346
 - equals, 346
 - getDataStructureType, 346
 - ID_ACTIVEMQOBJECTMESSAGE, 347
 - toString, 346
- activemq::commands::ActiveMQQueue, 379
 - ~ActiveMQQueue, 380
 - ActiveMQQueue, 380
 - clone, 380
 - cloneDataStructure, 380
 - copy, 380
 - copyDataStructure, 381
 - equals, 381
 - getCMSDestination, 381
 - getCMSProperties, 381
 - getDataStructureType, 381
 - getDestinationType, 382
 - getQueueName, 382
 - ID_ACTIVEMQQUEUE, 382
 - toString, 382
- activemq::commands::ActiveMQStreamMessage, 422
 - ~ActiveMQStreamMessage, 425
 - ActiveMQStreamMessage, 425
 - clearBody, 425
 - clone, 425
 - cloneDataStructure, 425
 - copyDataStructure, 425
 - equals, 426
 - getDataStructureType, 426
 - ID_ACTIVEMQSTREAMMESSAGE, 435
 - onSend, 426
 - readBoolean, 426
 - readByte, 427
 - readBytes, 427, 428
 - readChar, 428
 - readDouble, 429
 - readFloat, 429
 - readInt, 429
 - readLong, 430
 - readShort, 430
 - readString, 430
 - readUnsignedShort, 431
 - reset, 431
 - toString, 431
 - writeBoolean, 432
 - writeByte, 432
 - writeBytes, 432, 433
 - writeChar, 433
 - writeDouble, 433
 - writeFloat, 434

- writeInt, 434
- writeLong, 434
- writeShort, 434
- writeString, 435
- writeUnsignedShort, 435
- activemq::commands::ActiveMQTempDestination, 455
 - ~ActiveMQTempDestination, 457
 - ActiveMQTempDestination, 457
 - cloneDataStructure, 457
 - close, 457
 - connection, 458
 - copyDataStructure, 457
 - equals, 457
 - getDataStructureType, 458
 - ID_ACTIVEMQTEMPDESTINATION, 458
 - setConnection, 458
 - toString, 458
- activemq::commands::ActiveMQTempQueue, 477
 - ~ActiveMQTempQueue, 478
 - ActiveMQTempQueue, 478
 - clone, 478
 - cloneDataStructure, 478
 - copy, 479
 - copyDataStructure, 479
 - destroy, 479
 - equals, 479
 - getCMSDestination, 479
 - getCMSProperties, 480
 - getDataStructureType, 480
 - getDestinationType, 480
 - getQueueName, 480
 - ID_ACTIVEMQTEMPQUEUE, 481
 - toString, 480
- activemq::commands::ActiveMQTempTopic, 501
 - ~ActiveMQTempTopic, 502
 - ActiveMQTempTopic, 502
 - clone, 502
 - cloneDataStructure, 502
 - copy, 502
 - copyDataStructure, 503
 - destroy, 503
 - equals, 503
 - getCMSDestination, 503
 - getCMSProperties, 503
 - getDataStructureType, 504
 - getDestinationType, 504
 - getTopicName, 504
 - ID_ACTIVEMQTEMPTOPIC, 505
 - toString, 504
- activemq::commands::ActiveMQTextMessage, 525
 - ~ActiveMQTextMessage, 526
 - ActiveMQTextMessage, 526
 - beforeMarshal, 526
 - clearBody, 526
 - clone, 527
 - cloneDataStructure, 527
 - copyDataStructure, 527
 - equals, 527
 - getDataStructureType, 528
 - getSize, 528
 - getText, 528
 - ID_ACTIVEMQTEXTMESSAGE, 529
 - setText, 528, 529
 - text, 529
 - toString, 529
- activemq::commands::ActiveMQTopic, 549
 - ~ActiveMQTopic, 551
 - ActiveMQTopic, 551
 - clone, 551
 - cloneDataStructure, 551
 - copy, 551
 - copyDataStructure, 551
 - equals, 551
 - getCMSDestination, 552
 - getCMSProperties, 552
 - getDataStructureType, 552
 - getDestinationType, 552
 - getTopicName, 552
 - ID_ACTIVEMQTOPIC, 553
 - toString, 553
- activemq::commands::BaseCommand, 596
 - ~BaseCommand, 598
 - BaseCommand, 598
 - copyDataStructure, 598
 - equals, 598
 - getCommandId, 599
 - isBrokerInfo, 599
 - isConnectionInfo, 600
 - isConsumerInfo, 600
 - isKeepAliveInfo, 600
 - isMessage, 600
 - isMessageAck, 600
 - isMessageDispatch, 600
 - isMessageDispatchNotification, 600
 - isProducerAck, 601
 - isProducerInfo, 601
 - isRemoveInfo, 601
 - isRemoveSubscriptionInfo, 601
 - isResponse, 601
 - isResponseRequired, 601
 - isShutdownInfo, 602
 - isTransactionInfo, 602

- isWireFormatInfo, 602
- setCommandId, 602
- setResponseRequired, 602
- toString, 602
- activemq::commands::BaseDataStructure, 659
 - ~BaseDataStructure, 660
 - afterMarshal, 660
 - afterUnmarshal, 660
 - beforeMarshal, 660
 - beforeUnmarshal, 660
 - copyDataStructure, 661
 - equals, 661
 - getMarshaledForm, 661
 - isMarshalAware, 661
 - setMarshaledForm, 662
 - toString, 662
- activemq::commands::BooleanExpression, 679
 - ~BooleanExpression, 679
 - BooleanExpression, 679
 - cloneDataStructure, 679
 - copyDataStructure, 679
 - equals, 680
 - toString, 680
- activemq::commands::BrokerError, 686
 - ~BrokerError, 687
 - BrokerError, 687
 - cloneDataStructure, 687
 - copyDataStructure, 687
 - getCause, 687
 - getDataStructureType, 688
 - getExceptionClass, 688
 - getMessage, 688
 - getStackTraceElements, 688
 - setCause, 688
 - setExceptionClass, 689
 - setMessage, 689
 - setStackTraceElements, 689
 - visit, 689
- activemq::commands::BrokerError::StackTraceElement
 - 2811
 - ClassName, 2812
 - FileName, 2812
 - LineNumber, 2812
 - MethodName, 2812
- activemq::commands::BrokerId, 691
 - ~BrokerId, 692
 - BrokerId, 692
 - cloneDataStructure, 692
 - COMPARATOR, 692
 - compareTo, 692
 - copyDataStructure, 692
 - equals, 693
 - getDataStructureType, 693
 - getValue, 693
 - ID_BROKERID, 694
 - operator<, 693
 - operator=, 693
 - operator==, 693
 - setValue, 693
 - toString, 693
 - value, 694
- activemq::commands::BrokerInfo, 713
 - ~BrokerInfo, 715
 - brokerId, 721
 - BrokerInfo, 715
 - brokerName, 721
 - brokerUploadUrl, 721
 - brokerURL, 721
 - cloneDataStructure, 715
 - connectionId, 721
 - copyDataStructure, 715
 - duplexConnection, 721
 - equals, 716
 - faultTolerantConfiguration, 721
 - getBrokerId, 716, 717
 - getBrokerName, 717
 - getBrokerUploadUrl, 717
 - getBrokerURL, 717
 - getConnectionId, 717
 - getDataStructureType, 717
 - getNetworkProperties, 717, 718
 - getPeerBrokerInfos, 718
 - ID_BROKERINFO, 721
 - isBrokerInfo, 718
 - isDuplexConnection, 718
 - isFaultTolerantConfiguration, 719
 - isMasterBroker, 719
 - isNetworkConnection, 719
 - isSlaveBroker, 719
 - masterBroker, 721
 - networkConnection, 721
 - networkProperties, 721
 - operator=, 719
 - peerBrokerInfos, 721
 - setBrokerId, 719
 - setBrokerName, 719
 - setBrokerUploadUrl, 719
 - setBrokerURL, 719
 - setConnectionId, 719
 - setDuplexConnection, 719
 - setFaultTolerantConfiguration, 719
 - setMasterBroker, 719
 - setNetworkConnection, 719
 - setNetworkProperties, 719
 - setPeerBrokerInfos, 719
 - setSlaveBroker, 719
 - slaveBroker, 721
 - toString, 719

- visit, 720
- activemq::commands::Command, 991
 - ~Command, 992
 - getCommandId, 992
 - isBrokerInfo, 992
 - isConnectionInfo, 992
 - isConsumerInfo, 992
 - isKeepAliveInfo, 992
 - isMessage, 992
 - isMessageAck, 993
 - isMessageDispatch, 993
 - isMessageDispatchNotification, 993
 - isProducerAck, 993
 - isProducerInfo, 993
 - isRemoveInfo, 993
 - isRemoveSubscriptionInfo, 993
 - isResponse, 993
 - isResponseRequired, 994
 - isShutdownInfo, 994
 - isTransactionInfo, 994
 - isWireFormatInfo, 994
 - setCommandId, 994
 - setResponseRequired, 994
 - toString, 995
 - visit, 995
- activemq::commands::ConnectionControl, 1055
 - ~ConnectionControl, 1057
 - cloneDataStructure, 1057
 - close, 1059
 - ConnectionControl, 1057
 - copyDataStructure, 1057
 - equals, 1057
 - exit, 1059
 - faultTolerant, 1059
 - getDataStructureType, 1057
 - ID_CONNECTIONCONTROL, 1059
 - isClose, 1058
 - isExit, 1058
 - isFaultTolerant, 1058
 - isResume, 1058
 - isSuspend, 1058
 - operator=, 1058
 - resume, 1059
 - setClose, 1058
 - setExit, 1058
 - setFaultTolerant, 1058
 - setResume, 1058
 - setSuspend, 1058
 - suspend, 1059
 - toString, 1058
 - visit, 1058
- activemq::commands::ConnectionError, 1079
 - ~ConnectionError, 1081
 - cloneDataStructure, 1081
 - ConnectionError, 1081
 - connectionId, 1083
 - copyDataStructure, 1081
 - equals, 1081
 - exception, 1083
 - getConnectionId, 1081, 1082
 - getDataStructureType, 1082
 - getException, 1082
 - ID_CONNECTIONERROR, 1083
 - operator=, 1082
 - setConnectionId, 1082
 - setException, 1082
 - toString, 1082
 - visit, 1082
- activemq::commands::ConnectionId, 1106
 - ~ConnectionId, 1107
 - cloneDataStructure, 1107
 - COMPARATOR, 1107
 - compareTo, 1107
 - ConnectionId, 1107
 - copyDataStructure, 1107
 - equals, 1107, 1108
 - getDataStructureType, 1108
 - getValue, 1108
 - ID_CONNECTIONID, 1109
 - operator<, 1108
 - operator=, 1108
 - operator==, 1108
 - setValue, 1108
 - toString, 1108
 - value, 1109
- activemq::commands::ConnectionInfo, 1128
 - ~ConnectionInfo, 1130
 - brokerMasterConnector, 1134
 - brokerPath, 1134
 - clientId, 1134
 - clientMaster, 1134
 - cloneDataStructure, 1130
 - connectionId, 1134
 - ConnectionInfo, 1130
 - copyDataStructure, 1130
 - equals, 1130
 - getBrokerPath, 1131
 - getClientId, 1131
 - getConnectionId, 1131
 - getDataStructureType, 1131
 - getPassword, 1131, 1132
 - getUserName, 1132
 - ID_CONNECTIONINFO, 1134
 - isBrokerMasterConnector, 1132
 - isClientMaster, 1132
 - isConnectionInfo, 1132
 - isManageable, 1132
 - manageable, 1134

- operator=, 1133
- password, 1134
- setBrokerMasterConnector, 1133
- setBrokerPath, 1133
- setClientId, 1133
- setClientMaster, 1133
- setConnectionId, 1133
- setManageable, 1133
- setPassword, 1133
- setUserName, 1133
- toString, 1133
- userName, 1134
- visit, 1133
- activemq::commands::ConsumerControl, 1165
 - ~ConsumerControl, 1167
 - cloneDataStructure, 1167
 - close, 1170
 - ConsumerControl, 1167
 - consumerId, 1170
 - copyDataStructure, 1167
 - equals, 1167
 - flush, 1170
 - getConsumerId, 1168
 - getDataStructureType, 1168
 - getPrefetch, 1168
 - ID_CONSUMERCONTROL, 1170
 - isClose, 1169
 - isFlush, 1169
 - isStart, 1169
 - isStop, 1169
 - operator=, 1169
 - prefetch, 1170
 - setClose, 1169
 - setConsumerId, 1169
 - setFlush, 1169
 - setPrefetch, 1169
 - setStart, 1169
 - setStop, 1169
 - start, 1170
 - stop, 1170
 - toString, 1169
 - visit, 1169
- activemq::commands::ConsumerId, 1190
 - ~ConsumerId, 1192
 - cloneDataStructure, 1192
 - COMPARATOR, 1192
 - compareTo, 1192
 - connectionId, 1194
 - ConsumerId, 1192
 - copyDataStructure, 1192
 - equals, 1192, 1193
 - getConnectionId, 1193
 - getDataStructureType, 1193
 - getParentId, 1193
 - getSessionId, 1194
 - getValue, 1194
 - ID_CONSUMERID, 1194
 - operator<, 1194
 - operator=, 1194
 - operator==, 1194
 - sessionId, 1194
 - setConnectionId, 1194
 - setSessionId, 1194
 - setValue, 1194
 - toString, 1194
 - value, 1194
- activemq::commands::ConsumerInfo, 1214
 - ~ConsumerInfo, 1217
 - additionalPredicate, 1222
 - brokerPath, 1222
 - browser, 1222
 - cloneDataStructure, 1217
 - consumerId, 1222
 - ConsumerInfo, 1217
 - copyDataStructure, 1217
 - destination, 1222
 - dispatchAsync, 1222
 - equals, 1217
 - exclusive, 1222
 - getAdditionalPredicate, 1217, 1218
 - getBrokerPath, 1218
 - getConsumerId, 1218
 - getDataStructureType, 1218
 - getDestination, 1218, 1219
 - getMaximumPendingMessageLimit, 1219
 - getNetworkConsumerPath, 1219
 - getPrefetchSize, 1219
 - getPriority, 1219
 - getSelector, 1219
 - getSubscriptionName, 1219
 - ID_CONSUMERINFO, 1222
 - isBrowser, 1219
 - isConsumerInfo, 1219
 - isDispatchAsync, 1219
 - isExclusive, 1220
 - isNetworkSubscription, 1220
 - isNoLocal, 1220
 - isNoRangeAcks, 1220
 - isOptimizedAcknowledge, 1220
 - isRetroactive, 1220
 - maximumPendingMessageLimit, 1222
 - networkConsumerPath, 1222
 - networkSubscription, 1222
 - noLocal, 1222
 - noRangeAcks, 1222
 - operator=, 1220
 - optimizedAcknowledge, 1222
 - prefetchSize, 1222

- priority, 1222
- retroactive, 1222
- selector, 1222
- setAdditionalPredicate, 1220
- setBrokerPath, 1220
- setBrowser, 1220
- setConsumerId, 1220
- setDestination, 1220
- setDispatchAsync, 1220
- setExclusive, 1220
- setMaximumPendingMessageLimit, 1220
- setNetworkConsumerPath, 1220
- setNetworkSubscription, 1220
- setNoLocal, 1220
- setNoRangeAcks, 1220
- setOptimizedAcknowledge, 1220
- setPrefetchSize, 1220
- setPriority, 1220
- setRetroactive, 1220
- setSelector, 1220
- setSubscriptionName, 1220
- subscriptionName, 1222
- toString, 1220
- visit, 1221
- activemq::commands::ControlCommand, 1243
 - ~ControlCommand, 1244
 - cloneDataStructure, 1244
 - command, 1246
 - ControlCommand, 1244
 - copyDataStructure, 1244
 - equals, 1245
 - getCommand, 1245
 - getDataStructureType, 1245
 - ID_CONTROLCOMMAND, 1246
 - operator=, 1245
 - setCommand, 1245
 - toString, 1245
 - visit, 1246
- activemq::commands::DataArrayResponse, 1267
 - ~DataArrayResponse, 1269
 - cloneDataStructure, 1269
 - copyDataStructure, 1269
 - data, 1270
 - DataArrayResponse, 1269
 - equals, 1269
 - getData, 1269, 1270
 - getDataStructureType, 1270
 - ID_DATAARRAYRESPONSE, 1270
 - operator=, 1270
 - setData, 1270
 - toString, 1270
- activemq::commands::DataResponse, 1307
 - ~DataResponse, 1308
 - cloneDataStructure, 1308
 - copyDataStructure, 1308
 - data, 1310
 - DataResponse, 1308
 - equals, 1308
 - getData, 1309
 - getDataStructureType, 1309
 - ID_DATARESPONSE, 1310
 - operator=, 1309
 - setData, 1309
 - toString, 1309
- activemq::commands::DataStructure, 1372
 - ~DataStructure, 1373
 - cloneDataStructure, 1373
 - copyDataStructure, 1373
 - equals, 1374
 - getDataStructureType, 1375
 - toString, 1376
- activemq::commands::DestinationInfo, 1390
 - ~DestinationInfo, 1392
 - brokerPath, 1395
 - cloneDataStructure, 1392
 - connectionId, 1395
 - copyDataStructure, 1392
 - destination, 1395
 - DestinationInfo, 1392
 - equals, 1392
 - getBrokerPath, 1393
 - getConnectionId, 1393
 - getDataStructureType, 1393
 - getDestination, 1393, 1394
 - getOperationType, 1394
 - getTimeout, 1394
 - ID_DESTINATIONINFO, 1395
 - operationType, 1395
 - operator=, 1394
 - setBrokerPath, 1394
 - setConnectionId, 1394
 - setDestination, 1394
 - setOperationType, 1394
 - setTimeout, 1394
 - timeout, 1395
 - toString, 1394
 - visit, 1394
- activemq::commands::DiscoveryEvent, 1416
 - ~DiscoveryEvent, 1418
 - brokerName, 1420
 - cloneDataStructure, 1418
 - copyDataStructure, 1418
 - DiscoveryEvent, 1418
 - equals, 1418
 - getBrokerName, 1418, 1419
 - getDataStructureType, 1419
 - getServiceName, 1419

- ID_DISCOVERYEVENT, 1420
- operator=, 1419
- serviceName, 1420
- setBrokerName, 1419
- setServiceName, 1419
- toString, 1419
- activemq::commands::ExceptionResponse, 1484
 - ~ExceptionResponse, 1485
 - cloneDataStructure, 1485
 - copyDataStructure, 1485
 - equals, 1485
 - exception, 1486
 - ExceptionResponse, 1485
 - getDataStructureType, 1485
 - getException, 1486
 - ID_EXCEPTIONRESPONSE, 1486
 - operator=, 1486
 - setException, 1486
 - toString, 1486
- activemq::commands::FlushCommand, 1573
 - ~FlushCommand, 1574
 - cloneDataStructure, 1574
 - copyDataStructure, 1574
 - equals, 1574
 - FlushCommand, 1574
 - getDataStructureType, 1574
 - ID_FLUSHCOMMAND, 1575
 - operator=, 1575
 - toString, 1575
 - visit, 1575
- activemq::commands::IntegerResponse, 1667
 - ~IntegerResponse, 1668
 - cloneDataStructure, 1668
 - copyDataStructure, 1668
 - equals, 1668
 - getDataStructureType, 1668
 - getResult, 1669
 - ID_INTEGERRESPONSE, 1669
 - IntegerResponse, 1668
 - operator=, 1669
 - result, 1669
 - setResult, 1669
 - toString, 1669
- activemq::commands::JournalQueueAck, 1718
 - ~JournalQueueAck, 1719
 - cloneDataStructure, 1719
 - copyDataStructure, 1719
 - destination, 1721
 - equals, 1720
 - getDataStructureType, 1720
 - getDestination, 1720, 1721
 - getMessageAck, 1721
 - ID_JOURNALQUEUEACK, 1721
 - JournalQueueAck, 1719
 - messageAck, 1721
 - operator=, 1721
 - setDestination, 1721
 - setMessageAck, 1721
 - toString, 1721
- activemq::commands::JournalTopicAck, 1741
 - ~JournalTopicAck, 1743
 - clientId, 1746
 - cloneDataStructure, 1743
 - copyDataStructure, 1743
 - destination, 1746
 - equals, 1743
 - getClientId, 1743, 1744
 - getDataStructureType, 1744
 - getDestination, 1744, 1745
 - getMessageId, 1745
 - getMessageSequenceId, 1745
 - getSubscriptionName, 1745
 - getTransactionId, 1745
 - ID_JOURNALTOPICACK, 1746
 - JournalTopicAck, 1743
 - messageId, 1746
 - messageSequenceId, 1746
 - operator=, 1745
 - setClientId, 1745
 - setDestination, 1745
 - setMessageId, 1745
 - setMessageSequenceId, 1745
 - setSubscriptionName, 1745
 - setTransactionId, 1745
 - subscriptionName, 1746
 - toString, 1745
 - transactionId, 1746
- activemq::commands::JournalTrace, 1766
 - ~JournalTrace, 1767
 - cloneDataStructure, 1767
 - copyDataStructure, 1768
 - equals, 1768
 - getDataStructureType, 1768
 - getMessage, 1768, 1769
 - ID_JOURNALTRACE, 1769
 - JournalTrace, 1767
 - message, 1769
 - operator=, 1769
 - setMessage, 1769
 - toString, 1769
- activemq::commands::JournalTransaction, 1789
 - ~JournalTransaction, 1791
 - cloneDataStructure, 1791
 - copyDataStructure, 1791
 - equals, 1791
 - getDataStructureType, 1791
 - getTransactionId, 1792
 - getType, 1792

- getWasPrepared, 1792
- ID_JOURNALTRANSACTION, 1793
- JournalTransaction, 1791
- operator=, 1792
- setTransactionId, 1792
- setType, 1792
- setWasPrepared, 1792
- toString, 1792
- transactionId, 1793
- type, 1793
- wasPrepared, 1793
- activemq::commands::KeepAliveInfo, 1812
 - ~KeepAliveInfo, 1813
 - cloneDataStructure, 1813
 - copyDataStructure, 1814
 - equals, 1814
 - getDataStructureType, 1814
 - ID_KEEPLIVEINFO, 1815
 - isKeepAliveInfo, 1814
 - KeepAliveInfo, 1813
 - operator=, 1814
 - toString, 1815
 - visit, 1815
- activemq::commands::LastPartialCommand, 1840
 - ~LastPartialCommand, 1841
 - cloneDataStructure, 1841
 - copyDataStructure, 1841
 - equals, 1841
 - getDataStructureType, 1841
 - ID_LASTPARTIALCOMMAND, 1842
 - LastPartialCommand, 1841
 - operator=, 1842
 - toString, 1842
- activemq::commands::LocalTransactionId, 1874
 - ~LocalTransactionId, 1876
 - cloneDataStructure, 1876
 - COMPARATOR, 1876
 - compareTo, 1876
 - connectionId, 1878
 - copyDataStructure, 1876
 - equals, 1876
 - getConnectionId, 1877
 - getDataStructureType, 1877
 - getValue, 1877
 - ID_LOCALTRANSACTIONID, 1878
 - LocalTransactionId, 1876
 - operator<, 1877
 - operator=, 1877
 - operator==, 1877
 - setConnectionId, 1877
 - setValue, 1877
 - toString, 1877
 - value, 1878
- activemq::commands::Message, 2018
 - ~Message, 2022
 - afterUnmarshal, 2022
 - arrival, 2035
 - beforeMarshal, 2022
 - brokerInTime, 2035
 - brokerOutTime, 2035
 - brokerPath, 2035
 - cloneDataStructure, 2022
 - cluster, 2035
 - compressed, 2035
 - content, 2035
 - copyDataStructure, 2023
 - correlationId, 2035
 - dataStructure, 2035
 - DEFAULT_MESSAGE_SIZE, 2035
 - destination, 2035
 - droppable, 2035
 - equals, 2023
 - expiration, 2035
 - getAckHandler, 2024
 - getArrival, 2024
 - getBrokerInTime, 2025
 - getBrokerOutTime, 2025
 - getBrokerPath, 2025
 - getCluster, 2025
 - getContent, 2025
 - getCorrelationId, 2025
 - getDataStructure, 2025
 - getDataStructureType, 2025
 - getDestination, 2026, 2027
 - getExpiration, 2027
 - getGroupId, 2027
 - getGroupSequence, 2027
 - getMarshaledProperties, 2027
 - getMessageId, 2027
 - getMessageProperties, 2027
 - getOriginalDestination, 2027, 2028
 - getOriginalTransactionId, 2028
 - getPriority, 2028
 - getProducerId, 2028
 - getRedeliveryCounter, 2028
 - getReplyTo, 2028
 - getSize, 2028
 - getTargetConsumerId, 2028, 2029
 - getTimestamp, 2029
 - getTransactionId, 2029
 - getType, 2029
 - getUserID, 2029
 - groupId, 2035
 - groupSequence, 2035
 - ID_MESSAGE, 2035
 - isCompressed, 2029
 - isDroppable, 2029

- isExpired, 2029
- isMarshalAware, 2029
- isMessage, 2030
- isPersistent, 2030
- isReadOnlyBody, 2030
- isReadOnlyProperties, 2030
- isRecievedByDFBridge, 2030
- marshalledProperties, 2035
- Message, 2022
- messageId, 2035
- onSend, 2030
- operator=, 2030
- originalDestination, 2035
- originalTransactionId, 2035
- persistent, 2035
- priority, 2035
- producerId, 2035
- recievedByDFBridge, 2035
- redeliveryCounter, 2035
- replyTo, 2035
- setAckHandler, 2031
- setArrival, 2031
- setBrokerInTime, 2032
- setBrokerOutTime, 2032
- setBrokerPath, 2032
- setCluster, 2032
- setCompressed, 2032
- setContent, 2032
- setCorrelationId, 2032
- setDataStructure, 2032
- setDestination, 2032
- setDroppable, 2032
- setExpiration, 2032
- setGroupID, 2032
- setGroupSequence, 2032
- setMarshalledProperties, 2032
- setMessageId, 2032
- setOriginalDestination, 2032
- setOriginalTransactionId, 2032
- setPersistent, 2032
- setPriority, 2032
- setProducerId, 2032
- setReadOnlyBody, 2032
- setReadOnlyProperties, 2033
- setRecievedByDFBridge, 2033
- setRedeliveryCounter, 2033
- setReplyTo, 2033
- setTargetConsumerId, 2033
- setTimestamp, 2033
- setTransactionId, 2033
- setType, 2033
- setUserID, 2033
- targetConsumerId, 2035
- timestamp, 2035
- toString, 2033
- transactionId, 2035
- type, 2035
- userId, 2035
- visit, 2034
- activemq::commands::MessageAck, 2055
 - ~MessageAck, 2056
 - ackType, 2060
 - cloneDataStructure, 2056
 - consumerId, 2060
 - copyDataStructure, 2056
 - destination, 2060
 - equals, 2057
 - firstMessageId, 2060
 - getAckType, 2057
 - getConsumerId, 2057
 - getDataStructureType, 2057
 - getDestination, 2057, 2058
 - getFirstMessageId, 2058
 - getLastMessageId, 2058
 - getMessageCount, 2058
 - getTransactionId, 2058
 - ID_MESSAGEACK, 2060
 - isMessageAck, 2058
 - lastMessageId, 2060
 - MessageAck, 2056
 - messageCount, 2060
 - operator=, 2058
 - setAckType, 2059
 - setConsumerId, 2059
 - setDestination, 2059
 - setFirstMessageId, 2059
 - setLastMessageId, 2059
 - setMessageCount, 2059
 - setTransactionId, 2059
 - toString, 2059
 - transactionId, 2060
 - visit, 2059
- activemq::commands::MessageDispatch, 2084
 - ~MessageDispatch, 2085
 - cloneDataStructure, 2085
 - consumerId, 2088
 - copyDataStructure, 2085
 - destination, 2088
 - equals, 2085
 - getConsumerId, 2086
 - getDataStructureType, 2086
 - getDestination, 2086, 2087
 - getMessage, 2087
 - getRedeliveryCounter, 2087
 - ID_MESSAGEDISPATCH, 2088
 - isMessageDispatch, 2087
 - message, 2088
 - MessageDispatch, 2085

- operator=, 2087
- redeliveryCounter, 2088
- setConsumerId, 2087
- setDestination, 2087
- setMessage, 2087
- setRedeliveryCounter, 2087
- toString, 2087
- visit, 2088
- activemq::commands::MessageDispatchNotification, 2115
 - ~MessageDispatchNotification, 2117
 - cloneDataStructure, 2117
 - consumerId, 2120
 - copyDataStructure, 2117
 - deliverySequenceId, 2120
 - destination, 2120
 - equals, 2117
 - getConsumerId, 2118
 - getDataStructureType, 2118
 - getDeliverySequenceId, 2118
 - getDestination, 2118
 - getMessageId, 2118
 - ID_MESSAGEDISPATCHNOTIFICATION, 2120
 - isMessageDispatchNotification, 2118
 - MessageDispatchNotification, 2117
 - messageId, 2120
 - operator=, 2118
 - setConsumerId, 2119
 - setDeliverySequenceId, 2119
 - setDestination, 2119
 - setMessageId, 2119
 - toString, 2119
 - visit, 2119
- activemq::commands::MessageId, 2142
 - ~MessageId, 2143
 - brokerSequenceId, 2146
 - cloneDataStructure, 2143
 - COMPARATOR, 2143
 - compareTo, 2143
 - copyDataStructure, 2144
 - equals, 2144
 - getBrokerSequenceId, 2144
 - getDataStructureType, 2144
 - getProducerId, 2144, 2145
 - getProducerSequenceId, 2145
 - ID_MESSAGEID, 2146
 - MessageId, 2143
 - operator<, 2145
 - operator=, 2145
 - operator==, 2145
 - producerId, 2146
 - producerSequenceId, 2146
 - setBrokerSequenceId, 2145
 - setProducerId, 2145
 - setProducerSequenceId, 2145
 - toString, 2145
 - visit, 2145
- activemq::commands::MessagePull, 2202
 - ~MessagePull, 2204
 - cloneDataStructure, 2204
 - consumerId, 2207
 - copyDataStructure, 2204
 - correlationId, 2207
 - destination, 2207
 - equals, 2204
 - getConsumerId, 2204, 2205
 - getCorrelationId, 2205
 - getDataStructureType, 2205
 - getDestination, 2205, 2206
 - getMessageId, 2206
 - getTimeout, 2206
 - ID_MESSAGEPULL, 2207
 - messageId, 2207
 - MessagePull, 2204
 - operator=, 2206
 - setConsumerId, 2206
 - setCorrelationId, 2206
 - setDestination, 2206
 - setMessageId, 2206
 - setTimeout, 2206
 - timeout, 2207
 - toString, 2206
 - visit, 2206
- activemq::commands::NetworkBridgeFilter, 2246
 - ~NetworkBridgeFilter, 2247
 - cloneDataStructure, 2247
 - copyDataStructure, 2248
 - equals, 2248
 - getDataStructureType, 2248
 - getNetworkBrokerId, 2248, 2249
 - getNetworkTTL, 2249
 - ID_NETWORKBRIDGEFILTER, 2249
 - NetworkBridgeFilter, 2247
 - networkBrokerId, 2249
 - networkTTL, 2249
 - operator=, 2249
 - setNetworkBrokerId, 2249
 - setNetworkTTL, 2249
 - toString, 2249
- activemq::commands::PartialCommand, 2318
 - ~PartialCommand, 2320
 - cloneDataStructure, 2320
 - commandId, 2322
 - copyDataStructure, 2320
 - data, 2322
 - equals, 2320
 - getCommandId, 2320

- getData, 2321
- getDataStructureType, 2321
- ID_PARTIALCOMMAND, 2322
- operator=, 2321
- PartialCommand, 2320
- setCommandId, 2321
- setData, 2321
- toString, 2321
- activemq::commands::ProducerAck, 2420
 - ~ProducerAck, 2421
 - cloneDataStructure, 2421
 - copyDataStructure, 2421
 - equals, 2422
 - getDataStructureType, 2422
 - getProducerId, 2422
 - getSize, 2422
 - ID_PRODUCERACK, 2423
 - isProducerAck, 2422
 - operator=, 2422
 - ProducerAck, 2421
 - producerId, 2423
 - setProducerId, 2423
 - setSize, 2423
 - size, 2423
 - toString, 2423
 - visit, 2423
- activemq::commands::ProducerId, 2446
 - ~ProducerId, 2447
 - cloneDataStructure, 2448
 - COMPARATOR, 2447
 - compareTo, 2448
 - connectionId, 2449
 - copyDataStructure, 2448
 - equals, 2448
 - getConnectionId, 2448
 - getDataStructureType, 2448
 - getParentId, 2449
 - getSessionId, 2449
 - getValue, 2449
 - ID_PRODUCERID, 2449
 - operator<, 2449
 - operator=, 2449
 - operator==, 2449
 - ProducerId, 2447
 - sessionId, 2449
 - setConnectionId, 2449
 - setSessionId, 2449
 - setValue, 2449
 - toString, 2449
 - value, 2450
- activemq::commands::ProducerInfo, 2469
 - ~ProducerInfo, 2471
 - brokerPath, 2474
 - cloneDataStructure, 2471
 - copyDataStructure, 2471
 - destination, 2474
 - dispatchAsync, 2474
 - equals, 2471
 - getBrokerPath, 2472
 - getDataStructureType, 2472
 - getDestination, 2472
 - getProducerId, 2472
 - getWindowSize, 2472
 - ID_PRODUCERINFO, 2474
 - isDispatchAsync, 2472
 - isProducerInfo, 2472
 - operator=, 2472
 - producerId, 2474
 - ProducerInfo, 2471
 - setBrokerPath, 2473
 - setDestination, 2473
 - setDispatchAsync, 2473
 - setProducerId, 2473
 - setWindowSize, 2473
 - toString, 2473
 - visit, 2473
 - windowSize, 2474
- activemq::commands::RemoveInfo, 2536
 - ~RemoveInfo, 2538
 - cloneDataStructure, 2538
 - copyDataStructure, 2538
 - equals, 2538
 - getDataStructureType, 2538
 - getLastDeliveredSequenceId, 2539
 - getObjectId, 2539
 - ID_REMOVEINFO, 2540
 - isRemoveInfo, 2539
 - lastDeliveredSequenceId, 2540
 - objectId, 2540
 - operator=, 2539
 - RemoveInfo, 2538
 - setLastDeliveredSequenceId, 2539
 - setObjectId, 2539
 - toString, 2539
 - visit, 2539
- activemq::commands::RemoveSubscriptionInfo, 2560
 - ~RemoveSubscriptionInfo, 2561
 - clientId, 2564
 - cloneDataStructure, 2561
 - connectionId, 2564
 - copyDataStructure, 2561
 - equals, 2562
 - getClientId, 2562
 - getConnectionId, 2562
 - getDataStructureType, 2562
 - getSubscriptionName, 2562, 2563

- ID_REMOVE SUBSCRIPTIONINFO, 2564
- isRemoveSubscriptionInfo, 2563
- operator=, 2563
- RemoveSubscriptionInfo, 2561
- setClientId, 2563
- setConnectionId, 2563
- setSubscriptionName, 2563
- subscriptionName, 2564
- toString, 2563
- visit, 2563
- activemq::commands::ReplayCommand, 2584
 - ~ReplayCommand, 2585
 - cloneDataStructure, 2585
 - copyDataStructure, 2585
 - equals, 2585
 - firstNakNumber, 2587
 - getDataStructureType, 2586
 - getFirstNakNumber, 2586
 - getLastNakNumber, 2586
 - ID_REPLAYCOMMAND, 2587
 - lastNakNumber, 2587
 - operator=, 2586
 - ReplayCommand, 2585
 - setFirstNakNumber, 2586
 - setLastNakNumber, 2586
 - toString, 2586
 - visit, 2586
- activemq::commands::Response, 2611
 - ~Response, 2612
 - cloneDataStructure, 2612
 - copyDataStructure, 2612
 - correlationId, 2614
 - equals, 2612
 - getCorrelationId, 2613
 - getDataStructureType, 2613
 - ID_RESPONSE, 2614
 - isResponse, 2613
 - operator=, 2613
 - Response, 2612
 - setCorrelationId, 2614
 - toString, 2614
 - visit, 2614
- activemq::commands::SessionId, 2677
 - ~SessionId, 2679
 - cloneDataStructure, 2679
 - COMPARATOR, 2679
 - compareTo, 2679
 - connectionId, 2681
 - copyDataStructure, 2679
 - equals, 2679, 2680
 - getConnectionId, 2680
 - getDataStructureType, 2680
 - getParentId, 2680
 - getValue, 2680
 - ID_SESSIONID, 2681
 - operator<, 2680
 - operator=, 2681
 - operator==, 2681
 - SessionId, 2679
 - setConnectionId, 2681
 - setValue, 2681
 - toString, 2681
 - value, 2681
- activemq::commands::SessionInfo, 2701
 - ~SessionInfo, 2703
 - cloneDataStructure, 2703
 - copyDataStructure, 2703
 - equals, 2703
 - getAckMode, 2703
 - getDataStructureType, 2704
 - getSessionId, 2704
 - ID_SESSIONINFO, 2705
 - operator=, 2704
 - sessionId, 2705
 - SessionInfo, 2703
 - setAckMode, 2704
 - setSessionId, 2704
 - toString, 2704
 - visit, 2704
- activemq::commands::ShutdownInfo, 2757
 - ~ShutdownInfo, 2758
 - cloneDataStructure, 2758
 - copyDataStructure, 2758
 - equals, 2758
 - getDataStructureType, 2758
 - ID_SHUTDOWNINFO, 2759
 - isShutdownInfo, 2759
 - operator=, 2759
 - ShutdownInfo, 2758
 - toString, 2759
 - visit, 2759
- activemq::commands::SubscriptionInfo, 2907
 - ~SubscriptionInfo, 2908
 - clientId, 2911
 - cloneDataStructure, 2908
 - copyDataStructure, 2909
 - destination, 2911
 - equals, 2909
 - getClientId, 2909
 - getDataStructureType, 2909
 - getDestination, 2909, 2910
 - getSelector, 2910
 - getSubscriptionName, 2910
 - getSubscribedDestination, 2910
 - ID_SUBSCRIPTIONINFO, 2911
 - operator=, 2910
 - selector, 2911

- setClientId, 2910
- setDestination, 2910
- setSelector, 2910
- setSubscriptionName, 2910
- setSubscribedDestination, 2910
- subscriptionName, 2911
- subscribedDestination, 2911
- SubscriptionInfo, 2908
- toString, 2910
- activemq::commands::TransactionId, 3015
 - ~TransactionId, 3017
 - cloneDataStructure, 3017
 - COMPARATOR, 3017
 - compareTo, 3017
 - copyDataStructure, 3017
 - equals, 3017
 - getDataStructureType, 3018
 - ID_TRANSACTIONID, 3018
 - operator<, 3018
 - operator=, 3018
 - operator==, 3018
 - toString, 3018
 - TransactionId, 3017
- activemq::commands::TransactionInfo, 3037
 - ~TransactionInfo, 3038
 - cloneDataStructure, 3038
 - connectionId, 3040
 - copyDataStructure, 3038
 - equals, 3038
 - getConnectionId, 3039
 - getDataStructureType, 3039
 - getTransactionId, 3039
 - getType, 3039
 - ID_TRANSACTIONINFO, 3040
 - isTransactionInfo, 3039
 - operator=, 3039
 - setConnectionId, 3040
 - setTransactionId, 3040
 - setType, 3040
 - toString, 3040
 - transactionId, 3040
 - TransactionInfo, 3038
 - type, 3040
 - visit, 3040
- activemq::commands::WireFormatInfo, 3152
 - ~WireFormatInfo, 3155
 - afterUnmarshal, 3155
 - beforeMarshal, 3155
 - cloneDataStructure, 3155
 - copyDataStructure, 3156
 - equals, 3156
 - getCacheSize, 3156
 - getDataStructureType, 3156
 - getMagic, 3157
 - getMarshaledProperties, 3157
 - getMaxInactivityDuration, 3157
 - getMaxInactivityDurationInitialDelay, 3157
 - getProperties, 3157, 3158
 - getVersion, 3158
 - ID_WIREFORMATINFO, 3162
 - isCacheEnabled, 3158
 - isMarshalAware, 3158
 - isSizePrefixDisabled, 3158
 - isStackTraceEnabled, 3159
 - isTcpNoDelayEnabled, 3159
 - isTightEncodingEnabled, 3159
 - isValid, 3159
 - isWireFormatInfo, 3159
 - setCacheEnabled, 3159
 - setCacheSize, 3160
 - setMagic, 3160
 - setMarshaledProperties, 3160
 - setMaxInactivityDuration, 3160
 - setMaxInactivityDurationInitialDelay, 3160
 - setProperties, 3161
 - setSizePrefixDisabled, 3161
 - setStackTraceEnabled, 3161
 - setTcpNoDelayEnabled, 3161
 - setTightEncodingEnabled, 3161
 - setVersion, 3162
 - toString, 3162
 - visit, 3162
 - WireFormatInfo, 3155
- activemq::commands::XATransactionId, 3192
 - ~XATransactionId, 3194
 - branchQualifier, 3197
 - cloneDataStructure, 3194
 - COMPARATOR, 3194
 - compareTo, 3194
 - copyDataStructure, 3194
 - equals, 3194
 - formatId, 3197
 - getBranchQualifier, 3195
 - getDataStructureType, 3195
 - getFormatId, 3195
 - getGlobalTransactionId, 3196
 - globalTransactionId, 3197
 - ID_XATRANSACTIONID, 3197
 - operator<, 3196
 - operator=, 3196
 - operator==, 3196
 - setBranchQualifier, 3196
 - setFormatId, 3196
 - setGlobalTransactionId, 3196
 - toString, 3196
 - XATransactionId, 3194
- activemq::core, 72
- activemq::core::ActiveMQAckHandler, 141

- ~ActiveMQAckHandler, 141
- acknowledgeMessage, 141
- activemq::core::ActiveMQConnection, 202
 - ~ActiveMQConnection, 205
 - ActiveMQConnection, 204
 - addDispatcher, 205
 - addProducer, 205
 - addTransportListener, 205
 - close, 205
 - createSession, 206
 - destroyDestination, 206
 - fire, 207
 - getClientID, 207
 - getConnectionId, 207
 - getConnectionInfo, 207
 - getExceptionListener, 208
 - getMetaData, 208
 - isClosed, 208
 - isStarted, 208
 - onCommand, 208
 - oneway, 209
 - onException, 209
 - removeDispatcher, 209
 - removeProducer, 209
 - removeSession, 209
 - removeTransportListener, 210
 - sendPullRequest, 210
 - setExceptionListener, 210
 - start, 210
 - stop, 210
 - syncRequest, 211
 - transportInterrupted, 211
 - transportResumed, 211
- activemq::core::ActiveMQConnectionFactory, 211
 - ~ActiveMQConnectionFactory, 213
 - ActiveMQConnectionFactory, 213
 - createConnection, 213, 214
 - getBrokerURL, 215
 - getPassword, 215
 - getUsername, 215
 - setBrokerURL, 215
 - setPassword, 215
 - setUsername, 216
- activemq::core::ActiveMQConnectionMetaData, 216
 - ~ActiveMQConnectionMetaData, 217
 - ActiveMQConnectionMetaData, 217
 - getCMSMajorVersion, 217
 - getCMSMinorVersion, 217
 - getCMSProviderName, 218
 - getCMSVersion, 218
 - getCMSXPropertyNames, 218
 - getProviderMajorVersion, 219
 - getProviderMinorVersion, 219
 - getProviderVersion, 219
- activemq::core::ActiveMQConnectionSupport, 220
 - ~ActiveMQConnectionSupport, 222
 - ActiveMQConnectionSupport, 222
 - getClientId, 222
 - getCloseTimeout, 222
 - getNextLocalTransactionId, 222
 - getNextSessionId, 223
 - getNextTempDestinationId, 223
 - getPassword, 223
 - getProducerWindowSize, 223
 - getProperties, 223
 - getSendTimeout, 224
 - getTransport, 224
 - getUsername, 224
 - isAlwaysSyncSend, 224
 - isUseAsyncSend, 224
 - setAlwaysSyncSend, 225
 - setClientId, 225
 - setCloseTimeout, 225
 - setPassword, 225
 - setProducerWindowSize, 225
 - setSendTimeout, 226
 - setUseAsyncSend, 226
 - setUsername, 226
 - shutdownTransport, 226
 - startUpTransport, 226
- activemq::core::ActiveMQConstants, 227
 - ACK_TYPE_CONSUMED, 228
 - ACK_TYPE_DELIVERED, 228
 - ACK_TYPE_INDIVIDUAL, 228
 - ACK_TYPE_POISON, 228
 - ACK_TYPE_REDELIVERED, 228
 - AckType, 228
 - CONNECTION_ALWAYS_SYNC_SEND, 229
 - CONNECTION_CLOSE_TIMEOUT, 229
 - CONNECTION_PRODUCER_WINDOW_SIZE, 229
 - CONNECTION_SEND_TIMEOUT, 229
 - CONNECTION_USE_ASYNC_SEND, 229
 - CONSUMER_DISPATCH_ASYNC, 229
 - CONSUMER_EXCLUSIVE, 229
 - CONSUMER_NO_LOCAL, 229
 - CONSUMER_PREFETCH_SIZE, 229
 - CONSUMER_PRIORITY, 229
 - CONSUMER_RETROACTIVE, 229
 - CONSUMER_SELECTOR, 229
 - CUNSUMER_MAX_PENDING_MSG_LIMIT, 229
 - DESTINATION_ADD_OPERATION, 228

- DESTINATION_REMOVE_ - OPERATION, 228
- DestinationActions, 228
- DestinationOption, 228
- NUM_OPTIONS, 229
- NUM_PARAMS, 229
- PARAM_CLIENTID, 229
- PARAM_PASSWORD, 229
- PARAM_USERNAME, 229
- toDestinationOption, 230
- toString, 230
- toURIOption, 230
- TRANSACTION_STATE_BEGIN, 229
- TRANSACTION_STATE_ - COMMITONEPHASE, 229
- TRANSACTION_STATE_ - COMMITTWO PHASE, 229
- TRANSACTION_STATE_END, 229
- TRANSACTION_STATE_FORGET, 229
- TRANSACTION_STATE_PREPARE, 229
- TRANSACTION_STATE_RECOVER, 229
- TRANSACTION_STATE_ROLLBACK, 229
- TransactionState, 229
- URIParam, 229
- activemq::core::ActiveMQConstants::StaticInitializer, 2832
 - ~StaticInitializer, 2832
 - destOptionMap, 2832
 - destOptions, 2832
 - StaticInitializer, 2832
 - uriParams, 2832
 - uriParamsMap, 2832
- activemq::core::ActiveMQConsumer, 230
 - ~ActiveMQConsumer, 232
 - acknowledge, 232, 233
 - ActiveMQConsumer, 232
 - afterMessageIsConsumed, 233
 - beforeMessageIsConsumed, 233
 - clearMessagesInProgress, 233
 - close, 233
 - commit, 233
 - deliverAcks, 234
 - dequeue, 234
 - dispatch, 234
 - doClose, 234
 - getConsumerId, 234
 - getConsumerInfo, 235
 - getLastDeliveredSequenceId, 235
 - getMessageListener, 235
 - getMessageSelector, 235
 - isClosed, 235
 - isSynchronizationRegistered, 236
 - iterate, 236
 - receive, 236
 - receiveNoWait, 236
 - rollback, 237
 - setLastDeliveredSequenceId, 237
 - setMessageListener, 237
 - setSynchronizationRegistered, 237
 - start, 237
 - stop, 238
- activemq::core::ActiveMQProducer, 367
 - ~ActiveMQProducer, 369
 - ActiveMQProducer, 369
 - close, 369
 - getDeliveryMode, 369
 - getDisableMessageID, 369
 - getDisableMessageTimeStamp, 369
 - getPriority, 370
 - getProducerId, 370
 - getProducerInfo, 370
 - getSendTimeout, 370
 - getTimeToLive, 370
 - isClosed, 371
 - onProducerAck, 371
 - send, 371, 372
 - setDeliveryMode, 373
 - setDisableMessageID, 373
 - setDisableMessageTimeStamp, 373
 - setPriority, 374
 - setSendTimeout, 374
 - setTimeToLive, 374
- activemq::core::ActiveMQSession, 402
 - ~ActiveMQSession, 406
 - acknowledge, 406
 - ActiveMQSession, 406
 - ActiveMQSessionExecutor, 418
 - clearMessagesInProgress, 406
 - close, 407
 - commit, 407
 - createBrowser, 407
 - createBytesMessage, 408
 - createConsumer, 408, 409
 - createDurableConsumer, 409
 - createMapMessage, 410
 - createMessage, 410
 - createProducer, 410
 - createQueue, 410
 - createStreamMessage, 411
 - createTemporaryQueue, 411
 - createTemporaryTopic, 411
 - createTextMessage, 411, 412
 - createTopic, 412
 - deliverAcks, 412
 - dispatch, 412

- disposeOf, 413
- doStartTransaction, 413
- fire, 413
- getAcknowledgeMode, 414
- getConnection, 414
- getExceptionListener, 414
- getLastDeliveredSequenceId, 414
- getSessionId, 414
- getSessionInfo, 414
- isAutoAcknowledge, 415
- isClientAcknowledge, 415
- isDupsOkAcknowledge, 415
- isIndividualAcknowledge, 415
- isStarted, 415
- isTransacted, 415
- oneway, 415
- recover, 415
- redispatch, 416
- rollback, 416
- send, 416
- setLastDeliveredSequenceId, 417
- start, 417
- stop, 417
- syncRequest, 417
- unsubscribe, 417
- wakeup, 418
- activemq::core::ActiveMQSessionExecutor, 418
 - ~ActiveMQSessionExecutor, 419
 - ActiveMQSessionExecutor, 419
 - clear, 420
 - clearMessagesInProgress, 420
 - close, 420
 - execute, 420
 - executeFirst, 420
 - getUnconsumedMessages, 420
 - hasUnconsumedMessages, 420
 - isEmpty, 421
 - isRunning, 421
 - iterate, 421
 - start, 421
 - stop, 421
 - wakeup, 421
- activemq::core::ActiveMQTransactionContext, 573
 - ~ActiveMQTransactionContext, 574
 - ActiveMQTransactionContext, 574
 - addSynchronization, 574
 - begin, 574
 - commit, 575
 - getMaximumRedeliveries, 575
 - getRedeliveryDelay, 575
 - getTransactionId, 575
 - isInTransaction, 575
 - removeSynchronization, 576
 - rollback, 576
- activemq::core::DispatchData, 1439
 - DispatchData, 1440
 - getConsumerId, 1440
 - getMessage, 1440
- activemq::core::Dispatcher, 1440
 - ~Dispatcher, 1441
 - dispatch, 1441
- activemq::core::MessageDispatchChannel, 2088
 - ~MessageDispatchChannel, 2090
 - clear, 2090
 - close, 2090
 - dequeue, 2090
 - dequeueNoWait, 2091
 - enqueue, 2091
 - enqueueFirst, 2091
 - isClosed, 2091
 - isEmpty, 2091
 - isRunning, 2091
 - lock, 2092
 - MessageDispatchChannel, 2090
 - notify, 2092
 - notifyAll, 2092
 - peek, 2092
 - removeAll, 2093
 - size, 2093
 - start, 2093
 - stop, 2093
 - tryLock, 2093
 - unlock, 2093
 - wait, 2094, 2095
- activemq::core::Synchronization, 2945
 - ~Synchronization, 2946
 - afterCommit, 2946
 - afterRollback, 2946
 - beforeEnd, 2946
- activemq::exceptions, 73
- activemq::exceptions::ActiveMQException, 269
 - ~ActiveMQException, 271
 - ActiveMQException, 270
 - clone, 271
 - convertToCMSException, 271
- activemq::exceptions::BrokerException, 690
 - ~BrokerException, 690
 - BrokerException, 690
 - clone, 691
- activemq::io, 73
- activemq::io::LoggingInputStream, 1917
 - ~LoggingInputStream, 1917
 - LoggingInputStream, 1917
 - read, 1918
- activemq::io::LoggingOutputStream, 1919
 - ~LoggingOutputStream, 1919
 - LoggingOutputStream, 1919

- write, 1919, 1920
- activemq::library, 73
- activemq::library::ActiveMQCPP, 238
 - ~ActiveMQCPP, 239
 - ActiveMQCPP, 239
 - initializeLibrary, 239
 - operator=, 239
 - shutdownLibrary, 239
- activemq::state, 73
- activemq::state::CommandVisitor, 996
 - ~CommandVisitor, 998
 - processBeginTransaction, 998
 - processBrokerError, 998
 - processBrokerInfo, 999
 - processCommitTransactionOnePhase, 999
 - processCommitTransactionTwoPhase, 999
 - processConnectionControl, 999
 - processConnectionError, 999
 - processConnectionInfo, 999
 - processConsumerControl, 999
 - processConsumerInfo, 999
 - processControlCommand, 999
 - processDestinationInfo, 1000
 - processEndTransaction, 1000
 - processFlushCommand, 1000
 - processForgetTransaction, 1000
 - processKeepAliveInfo, 1000
 - processMessage, 1000
 - processMessageAck, 1000
 - processMessageDispatch, 1000
 - processMessageDispatchNotification, 1001
 - processMessagePull, 1001
 - processPrepareTransaction, 1001
 - processProducerAck, 1001
 - processProducerInfo, 1001
 - processRecoverTransactions, 1001
 - processRemoveConnection, 1001
 - processRemoveConsumer, 1001
 - processRemoveDestination, 1001
 - processRemoveInfo, 1002
 - processRemoveProducer, 1002
 - processRemoveSession, 1002
 - processRemoveSubscriptionInfo, 1002
 - processReplayCommand, 1002
 - processResponse, 1002
 - processRollbackTransaction, 1002
 - processSessionInfo, 1002
 - processShutdownInfo, 1003
 - processTransactionInfo, 1003
 - processWireFormat, 1003
- activemq::state::CommandVisitorAdapter, 1003
 - ~CommandVisitorAdapter, 1007
 - processBeginTransaction, 1007
 - processBrokerError, 1007
 - processBrokerInfo, 1007
 - processCommitTransactionOnePhase, 1007
 - processCommitTransactionTwoPhase, 1007
 - processConnectionControl, 1007
 - processConnectionError, 1007
 - processConnectionInfo, 1007
 - processConsumerControl, 1007
 - processConsumerInfo, 1007
 - processControlCommand, 1007
 - processDestinationInfo, 1007
 - processEndTransaction, 1007
 - processFlushCommand, 1007
 - processForgetTransaction, 1007
 - processKeepAliveInfo, 1007
 - processMessage, 1007
 - processMessageAck, 1007
 - processMessageDispatch, 1007
 - processMessageDispatchNotification, 1007
 - processMessagePull, 1007
 - processPrepareTransaction, 1007
 - processProducerAck, 1007
 - processProducerInfo, 1007
 - processRecoverTransactions, 1007
 - processRemoveConnection, 1007
 - processRemoveConsumer, 1007
 - processRemoveDestination, 1007
 - processRemoveInfo, 1007
 - processRemoveProducer, 1008
 - processRemoveSession, 1008
 - processRemoveSubscriptionInfo, 1008
 - processReplayCommand, 1008
 - processResponse, 1008
 - processRollbackTransaction, 1008
 - processSessionInfo, 1008
 - processShutdownInfo, 1008
 - processTransactionInfo, 1008
 - processWireFormat, 1009
- activemq::state::ConnectionState, 1158
 - ~ConnectionState, 1159
 - addSession, 1159
 - addTempDestination, 1159
 - addTransactionState, 1159
 - checkShutdown, 1159
 - ConnectionState, 1159
 - getInfo, 1159
 - getSessionState, 1159
 - getSessionStates, 1159
 - getTempDestinations, 1159
 - getTransactionState, 1159
 - getTransactionStates, 1159
 - removeSession, 1159
 - removeTempDestination, 1159
 - removeTransactionState, 1159

- reset, 1160
- shutdown, 1160
- toString, 1160
- activemq::state::ConnectionStateTracker, 1160
 - ~ConnectionStateTracker, 1162
 - ConnectionStateTracker, 1162
 - getMaxCacheSize, 1162
 - isRestoreConsumers, 1162
 - isRestoreProducers, 1162
 - isRestoreSessions, 1162
 - isRestoreTransaction, 1162
 - isTrackMessages, 1162
 - isTrackTransactions, 1162
 - processBeginTransaction, 1162
 - processCommitTransactionOnePhase, 1162
 - processCommitTransactionTwoPhase, 1162
 - processConnectionInfo, 1162
 - processConsumerInfo, 1163
 - processDestinationInfo, 1163
 - processEndTransaction, 1163
 - processMessage, 1163
 - processMessageAck, 1163
 - processPrepareTransaction, 1163
 - processProducerInfo, 1163
 - processRemoveConnection, 1164
 - processRemoveConsumer, 1164
 - processRemoveDestination, 1164
 - processRemoveProducer, 1164
 - processRemoveSession, 1164
 - processRollbackTransaction, 1164
 - processSessionInfo, 1164
 - RemoveTransactionAction, 1165
 - restore, 1165
 - setMaxCacheSize, 1165
 - setRestoreConsumers, 1165
 - setRestoreProducers, 1165
 - setRestoreSessions, 1165
 - setRestoreTransaction, 1165
 - setTrackMessages, 1165
 - setTrackTransactions, 1165
 - track, 1165
 - trackBack, 1165
- activemq::state::ConsumerState, 1242
 - ~ConsumerState, 1243
 - ConsumerState, 1243
 - getInfo, 1243
 - toString, 1243
- activemq::state::ProducerState, 2494
 - ~ProducerState, 2494
 - getInfo, 2494
 - ProducerState, 2494
 - toString, 2494
- activemq::state::SessionState, 2726
 - ~SessionState, 2728
 - addConsumer, 2728
 - addProducer, 2728
 - checkShutdown, 2728
 - getConsumerState, 2728
 - getConsumerStates, 2728
 - getInfo, 2728
 - getProducerState, 2728
 - getProducerStates, 2728
 - removeConsumer, 2728
 - removeProducer, 2728
 - SessionState, 2728
 - shutdown, 2728
 - toString, 2728
- activemq::state::Tracked, 3015
 - ~Tracked, 3015
 - isWaitingForResponse, 3015
 - onResponse, 3015
 - Tracked, 3015
- activemq::state::TransactionState, 3060
 - ~TransactionState, 3061
 - addCommand, 3061
 - checkShutdown, 3061
 - getCommands, 3061
 - getId, 3061
 - getPreparedResult, 3061
 - isPrepared, 3061
 - setPrepared, 3061
 - setPreparedResult, 3061
 - shutdown, 3061
 - toString, 3061
 - TransactionState, 3061
- activemq::threads, 74
- activemq::threads::CompositeTask, 1016
 - ~CompositeTask, 1016
 - isPending, 1016
- activemq::threads::CompositeTaskRunner, 1017
 - ~CompositeTaskRunner, 1018
 - addTask, 1018
 - CompositeTaskRunner, 1018
 - iterate, 1018
 - removeTask, 1018
 - run, 1018
 - shutdown, 1018
 - wakeup, 1018
- activemq::threads::DedicatedTaskRunner, 1382
 - ~DedicatedTaskRunner, 1383
 - DedicatedTaskRunner, 1383
 - run, 1383
 - shutdown, 1383
 - wakeup, 1383
- activemq::threads::Task, 2956
 - ~Task, 2956
 - iterate, 2956

- activemq::threads::TaskRunner, 2958
 - ~TaskRunner, 2958
 - shutdown, 2958
 - wakeup, 2959
- activemq::transport, 74
- activemq::transport::AbstractTransportFactory, 139
 - ~AbstractTransportFactory, 140
 - createWireFormat, 140
- activemq::transport::CompositeTransport, 1019
 - ~CompositeTransport, 1020
 - addURI, 1020
 - removeURI, 1020
- activemq::transport::correlator, 75
- activemq::transport::correlator::FutureResponse, 1600
 - ~FutureResponse, 1601
 - FutureResponse, 1601
 - getResponse, 1601
 - setResponse, 1601
- activemq::transport::correlator::ResponseCorrelator, 2616
 - ~ResponseCorrelator, 2617
 - close, 2617
 - onCommand, 2617
 - oneway, 2618
 - onTransportException, 2618
 - request, 2618, 2619
 - ResponseCorrelator, 2617
 - start, 2619
- activemq::transport::DefaultTransportListener, 1384
 - ~DefaultTransportListener, 1384
 - onCommand, 1384
 - onException, 1385
 - transportInterrupted, 1385
 - transportResumed, 1385
- activemq::transport::failover, 75
- activemq::transport::failover::BackupTransport, 591
 - ~BackupTransport, 592
 - BackupTransport, 592
 - getTransport, 592
 - getUri, 592
 - isClosed, 593
 - onException, 593
 - setClosed, 593
 - setTransport, 593
 - setUri, 593
- activemq::transport::failover::BackupTransportPool, 594
 - ~BackupTransportPool, 595
 - BackupTransport, 596
 - BackupTransportPool, 595
 - getBackup, 595
 - getBackupPoolSize, 595
 - isEnabled, 595
 - isPending, 595
 - iterate, 596
 - setBackupPoolSize, 596
 - setEnabled, 596
- activemq::transport::failover::CloseTransportsTask, 952
 - ~CloseTransportsTask, 953
 - add, 953
 - CloseTransportsTask, 953
 - isPending, 953
 - iterate, 953
- activemq::transport::failover::FailoverTransport, 1512
 - ~FailoverTransport, 1515
 - add, 1515
 - addURI, 1515
 - close, 1515
 - FailoverTransport, 1515
 - FailoverTransportListener, 1522
 - getBackOffMultiplier, 1516
 - getBackupPoolSize, 1516
 - getInitialReconnectDelay, 1516
 - getMaxCacheSize, 1516
 - getMaxReconnectAttempts, 1516
 - getMaxReconnectDelay, 1516
 - getReconnectDelay, 1516
 - getRemoteAddress, 1516
 - getTimeout, 1516
 - getTransportListener, 1516
 - handleTransportFailure, 1516
 - isBackup, 1517
 - isClosed, 1517
 - isConnected, 1517
 - isFaultTolerant, 1517
 - isInitialized, 1517
 - isPending, 1518
 - isRandomize, 1518
 - isTrackMessages, 1518
 - isUseExponentialBackOff, 1518
 - iterate, 1518
 - narrow, 1518
 - oneway, 1519
 - reconnect, 1519
 - removeURI, 1519
 - request, 1519, 1520
 - restoreTransport, 1520
 - setBackOffMultiplier, 1520
 - setBackup, 1521
 - setBackupPoolSize, 1521
 - setInitialized, 1521
 - setInitialReconnectDelay, 1521

- setMaxCacheSize, 1521
- setMaxReconnectAttempts, 1521
- setMaxReconnectDelay, 1521
- setRandomize, 1521
- setReconnectDelay, 1521
- setTimeout, 1521
- setTrackMessages, 1521
- setTransportListener, 1521
- setUseExponentialBackOff, 1522
- setWireFormat, 1522
- start, 1522
- stop, 1522
- activemq::transport::failover::FailoverTransportFactory, 1523
 - ~FailoverTransportFactory, 1524
 - create, 1524
 - createComposite, 1524
 - doCreateComposite, 1524
- activemq::transport::failover::FailoverTransportListener, 1525
 - ~FailoverTransportListener, 1526
 - FailoverTransportListener, 1526
 - onCommand, 1526
 - onException, 1526
 - transportInterrupted, 1526
 - transportResumed, 1526
- activemq::transport::failover::URIPool, 3117
 - ~URIPool, 3118
 - addURI, 3118
 - addURIs, 3118
 - getURI, 3118
 - isRandomize, 3119
 - removeURI, 3119
 - setRandomize, 3119
 - URIPool, 3118
- activemq::transport::inactivity, 76
- activemq::transport::inactivity::InactivityMonitor, 1624
 - ~InactivityMonitor, 1625
 - AsyncSignalReadErrorTask, 1627
 - AsyncWriteTask, 1627
 - close, 1625
 - getInitialDelayTime, 1625
 - getReadCheckTime, 1626
 - getWriteCheckTime, 1626
 - InactivityMonitor, 1625
 - isKeepAliveResponseRequired, 1626
 - onCommand, 1626
 - oneway, 1626
 - onException, 1626
 - ReadChecker, 1627
 - setInitialDelayTime, 1627
 - setKeepAliveResponseRequired, 1627
 - setReadCheckTime, 1627
 - setWriteCheckTime, 1627
 - WriteChecker, 1627
- activemq::transport::inactivity::ReadChecker, 2517
 - ~ReadChecker, 2518
 - ReadChecker, 2518
 - run, 2518
- activemq::transport::inactivity::WriteChecker, 3186
 - ~WriteChecker, 3186
 - run, 3186
 - WriteChecker, 3186
- activemq::transport::IOTransport, 1709
 - ~IOTransport, 1711
 - close, 1711
 - getRemoteAddress, 1711
 - getTransportListener, 1712
 - IOTransport, 1711
 - isClosed, 1712
 - isConnected, 1712
 - isFaultTolerant, 1712
 - narrow, 1712
 - oneway, 1713
 - reconnect, 1713
 - request, 1713
 - run, 1714
 - setInputStream, 1714
 - setOutputStream, 1714
 - setTransportListener, 1714
 - setWireFormat, 1714
 - start, 1715
 - stop, 1715
- activemq::transport::logging, 76
- activemq::transport::logging::LoggingTransport, 1920
 - ~LoggingTransport, 1921
 - LoggingTransport, 1921
 - onCommand, 1921
 - oneway, 1921
 - request, 1922
- activemq::transport::mock, 76
- activemq::transport::mock::InternalCommandListener, 1689
 - ~InternalCommandListener, 1690
 - InternalCommandListener, 1690
 - onCommand, 1690
 - run, 1690
 - setResponseBuilder, 1691
 - setTransport, 1691
- activemq::transport::mock::MockTransport, 2227
 - ~MockTransport, 2229
 - close, 2229
 - fireCommand, 2230

- fireException, 2230
- getInstance, 2230
- getNumReceivedMessageBeforeFail, 2231
- getNumReceivedMessages, 2231
- getNumSentKeepAlives, 2231
- getNumSentKeepAlivesBeforeFail, 2231
- getNumSentMessageBeforeFail, 2231
- getNumSentMessages, 2231
- getRemoteAddress, 2231
- getTransportListener, 2231
- getWireFormat, 2231
- isClosed, 2232
- isConnected, 2232
- isFailOnClose, 2232
- isFailOnKeepAliveSends, 2232
- isFailOnReceiveMessage, 2232
- isFailOnSendMessage, 2232
- isFailOnStart, 2232
- isFailOnStop, 2232
- isFaultTolerant, 2232
- MockTransport, 2229
- narrow, 2233
- oneway, 2233
- reconnect, 2233
- request, 2233, 2234
- setFailOnClose, 2234
- setFailOnKeepAliveSends, 2235
- setFailOnReceiveMessage, 2235
- setFailOnSendMessage, 2235
- setFailOnStart, 2235
- setFailOnStop, 2235
- setNumReceivedMessageBeforeFail, 2235
- setNumReceivedMessages, 2235
- setNumSentKeepAlives, 2235
- setNumSentKeepAlivesBeforeFail, 2235
- setNumSentMessageBeforeFail, 2235
- setNumSentMessages, 2235
- setOutgoingListener, 2235
- setResponseBuilder, 2235
- setTransportListener, 2236
- setWireFormat, 2236
- start, 2236
- stop, 2236
- activemq::transport::mock::MockTransportFactory, 2237
 - ~MockTransportFactory, 2238
 - create, 2238
 - createComposite, 2238
 - doCreateComposite, 2238
- activemq::transport::mock::ResponseBuilder, 2614
 - ~ResponseBuilder, 2615
 - buildIncomingCommands, 2615
 - buildResponse, 2615
- activemq::transport::tcp, 77
- activemq::transport::tcp::TcpTransport, 2967
 - ~TcpTransport, 2968
 - close, 2968
 - isClosed, 2968
 - isConnected, 2968
 - isFaultTolerant, 2969
 - TcpTransport, 2968
- activemq::transport::tcp::TcpTransportFactory, 2969
 - ~TcpTransportFactory, 2970
 - create, 2970
 - createComposite, 2970
 - doCreateComposite, 2971
- activemq::transport::Transport, 3066
 - ~Transport, 3067
 - getRemoteAddress, 3067
 - getTransportListener, 3067
 - isClosed, 3068
 - isConnected, 3068
 - isFaultTolerant, 3068
 - narrow, 3068
 - oneway, 3069
 - reconnect, 3069
 - request, 3069, 3070
 - setTransportListener, 3070
 - setWireFormat, 3070
 - start, 3070
 - stop, 3071
- activemq::transport::TransportFactory, 3071
 - ~TransportFactory, 3072
 - create, 3072
 - createComposite, 3072
- activemq::transport::TransportFilter, 3073
 - ~TransportFilter, 3075
 - close, 3075
 - fire, 3075
 - getRemoteAddress, 3076
 - getTransportListener, 3076
 - isClosed, 3076
 - isConnected, 3076
 - isFaultTolerant, 3076
 - listener, 3080
 - narrow, 3077
 - next, 3080
 - onCommand, 3077
 - oneway, 3077
 - onException, 3078
 - reconnect, 3078
 - request, 3078, 3079
 - setTransportListener, 3079
 - setWireFormat, 3079
 - start, 3079
 - stop, 3080

- TransportFilter, 3075
- transportInterrupted, 3080
- transportResumed, 3080
- activemq::transport::TransportListener, 3080
 - ~TransportListener, 3081
 - onCommand, 3081
 - onException, 3081
 - transportInterrupted, 3082
 - transportResumed, 3082
- activemq::transport::TransportRegistry, 3082
 - ~TransportRegistry, 3083
 - findFactory, 3083
 - getInstance, 3084
 - getTransportNames, 3084
 - registerFactory, 3084
 - unregisterFactory, 3084
- activemq::util, 77
- activemq::util::ActiveMQProperties, 374
 - ~ActiveMQProperties, 376
 - clear, 376
 - clone, 376
 - copy, 376
 - getProperties, 376, 377
 - getProperty, 377
 - hasProperty, 377
 - isEmpty, 377
 - remove, 378
 - setProperties, 378
 - setProperty, 378
 - toArray, 378
 - toString, 378
- activemq::util::CMSExceptionSupport, 963
 - ~CMSExceptionSupport, 964
 - create, 964
 - createMessageEOFException, 964
 - createMessageFormatException, 964
- activemq::util::CompositeData, 1013
 - ~CompositeData, 1015
 - CompositeData, 1015
 - getComponents, 1015
 - getFragment, 1015
 - getHost, 1015
 - getParameters, 1015
 - getPath, 1015
 - getScheme, 1015
 - setComponents, 1015
 - setFragment, 1015
 - setHost, 1015
 - setParameters, 1015
 - setPath, 1015
 - setScheme, 1015
 - toURI, 1015
- activemq::util::LongSequenceGenerator, 1967
 - ~LongSequenceGenerator, 1967
 - getLastSequenceId, 1967
 - getNextSequenceId, 1967
 - LongSequenceGenerator, 1967
- activemq::util::MemoryUsage, 2014
 - ~MemoryUsage, 2016
 - decreaseUsage, 2016
 - enqueueUsage, 2016
 - getLimit, 2016
 - getUsage, 2016
 - increaseUsage, 2016
 - isFull, 2017
 - MemoryUsage, 2015
 - setLimit, 2017
 - setUsage, 2017
 - waitForSpace, 2017
- activemq::util::PrimitiveList, 2370
 - ~PrimitiveList, 2373
 - getBool, 2373
 - getByte, 2374
 - getByteArray, 2374
 - getChar, 2374
 - getDouble, 2375
 - getFloat, 2375
 - getInt, 2376
 - getLong, 2376
 - getShort, 2376
 - getString, 2377
 - PrimitiveList, 2373
 - setBool, 2377
 - setByte, 2377
 - setByteArray, 2378
 - setChar, 2378
 - setDouble, 2378
 - setFloat, 2379
 - setInt, 2379
 - setLong, 2379
 - setShort, 2380
 - setString, 2380
 - toString, 2380
- activemq::util::PrimitiveMap, 2381
 - ~PrimitiveMap, 2383
 - getBool, 2384
 - getByte, 2384
 - getByteArray, 2384
 - getChar, 2385
 - getDouble, 2385
 - getFloat, 2386
 - getInt, 2386
 - getLong, 2386
 - getShort, 2387
 - getString, 2387
 - PrimitiveMap, 2383
 - setBool, 2388
 - setByte, 2388

- setByteArray, 2388
- setChar, 2388
- setDouble, 2389
- setFloat, 2389
- setInt, 2389
- setLong, 2389
- setShort, 2389
- setString, 2390
- toString, 2390
- activemq::util::PrimitiveValueConverter, 2397
 - ~PrimitiveValueConverter, 2397
 - convert, 2397
 - PrimitiveValueConverter, 2397
- activemq::util::PrimitiveValueNode, 2398
 - ~PrimitiveValueNode, 2404
 - BIG_STRING_TYPE, 2402
 - BOOLEAN_TYPE, 2401
 - BYTE_ARRAY_TYPE, 2402
 - BYTE_TYPE, 2402
 - CHAR_TYPE, 2402
 - clear, 2405
 - DOUBLE_TYPE, 2402
 - FLOAT_TYPE, 2402
 - getBool, 2405
 - getByte, 2405
 - getByteArray, 2405
 - getChar, 2405
 - getDouble, 2406
 - getFloat, 2406
 - getInt, 2406
 - getList, 2406
 - getLong, 2407
 - getMap, 2407
 - getShort, 2407
 - getString, 2407
 - getType, 2408
 - getValue, 2408
 - INTEGER_TYPE, 2402
 - LIST_TYPE, 2402
 - LONG_TYPE, 2402
 - MAP_TYPE, 2402
 - NULL_TYPE, 2401
 - operator=, 2408
 - operator==, 2408
 - PrimitiveType, 2401
 - PrimitiveValueNode, 2402–2404
 - setBool, 2408
 - setByte, 2409
 - setByteArray, 2409
 - setChar, 2409
 - setDouble, 2409
 - setFloat, 2409
 - setInt, 2410
 - setList, 2410
 - setLong, 2410
 - setMap, 2410
 - setShort, 2410
 - setString, 2411
 - setValue, 2411
 - SHORT_TYPE, 2402
 - STRING_TYPE, 2402
 - toString, 2411
- activemq::util::PrimitiveValueNode::PrimitiveValue, 2395
 - boolValue, 2396
 - byteArrayValue, 2396
 - byteValue, 2396
 - charValue, 2396
 - doubleValue, 2396
 - floatValue, 2396
 - intValue, 2396
 - listValue, 2396
 - longValue, 2396
 - mapValue, 2396
 - shortValue, 2396
 - stringValue, 2396
- activemq::util::URISupport, 3119
 - createQueryString, 3120
 - parseComposite, 3120
 - parseQuery, 3121
 - parseURL, 3121
- activemq::util::Usage, 3137
 - ~Usage, 3137
 - decreaseUsage, 3137
 - enqueueUsage, 3137
 - increaseUsage, 3138
 - isFull, 3138
 - waitForSpace, 3138
- activemq::wireformat, 78
- activemq::wireformat::MarshalAware, 1992
 - ~MarshalAware, 1993
 - afterMarshal, 1993
 - afterUnmarshal, 1993
 - beforeMarshal, 1994
 - beforeUnmarshal, 1994
 - getMarshaledForm, 1994
 - isMarshalAware, 1994
 - setMarshaledForm, 1995
- activemq::wireformat::openwire, 78
- activemq::wireformat::openwire::marshal, 78
- activemq::wireformat::openwire::marshal::BaseDataStreamMarshal, 636
 - ~BaseDataStreamMarshaller, 643
 - looseMarshal, 643
 - looseMarshalBrokerError, 643
 - looseMarshalCachedObject, 643
 - looseMarshalLong, 644
 - looseMarshalNestedObject, 644

- looseMarshalObjectArray, 644
- looseMarshalString, 645
- looseUnmarshal, 645
- looseUnmarshalBrokerError, 646
- looseUnmarshalByteArray, 646
- looseUnmarshalCachedObject, 646
- looseUnmarshalConstByteArray, 647
- looseUnmarshalLong, 647
- looseUnmarshalNestedObject, 647
- looseUnmarshalString, 648
- readAsciiString, 648
- tightMarshal1, 648
- tightMarshal2, 649
- tightMarshalBrokerError1, 649
- tightMarshalBrokerError2, 650
- tightMarshalCachedObject1, 650
- tightMarshalCachedObject2, 650
- tightMarshalLong1, 651
- tightMarshalLong2, 651
- tightMarshalNestedObject1, 652
- tightMarshalNestedObject2, 652
- tightMarshalObjectArray1, 652
- tightMarshalObjectArray2, 653
- tightMarshalString1, 653
- tightMarshalString2, 654
- tightUnmarshal, 654
- tightUnmarshalBrokerError, 655
- tightUnmarshalByteArray, 655
- tightUnmarshalCachedObject, 655
- tightUnmarshalConstByteArray, 656
- tightUnmarshalLong, 656
- tightUnmarshalNestedObject, 657
- tightUnmarshalString, 657
- toHexFromBytes, 657
- toString, 658
- activemq::wireformat::openwire::marshal::DataStreamMarshaller, 1330
 - ~DataStreamMarshaller, 1331
 - createObject, 1331
 - getDataStructureType, 1336
 - looseMarshal, 1342
 - looseUnmarshal, 1348
 - tightMarshal1, 1354
 - tightMarshal2, 1360
 - tightUnmarshal, 1366
- activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller, 2390
 - ~PrimitiveTypesMarshaller, 2392
 - marshal, 2392
 - marshalPrimitive, 2392
 - marshalPrimitiveList, 2393
 - marshalPrimitiveMap, 2393
 - PrimitiveTypesMarshaller, 2392
 - unmarshal, 2393, 2394
 - unmarshalPrimitive, 2394
 - unmarshalPrimitiveList, 2394
 - unmarshalPrimitiveMap, 2395
- activemq::wireformat::openwire::marshal::v1, 79
- activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller, 150
 - ~ActiveMQBlobMessageMarshaller, 152
 - ActiveMQBlobMessageMarshaller, 152
 - createObject, 152
 - getDataStructureType, 152
 - looseMarshal, 152
 - looseUnmarshal, 152
 - tightMarshal1, 153
 - tightMarshal2, 153
 - tightUnmarshal, 154
- activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller, 186
 - ~ActiveMQBytesMessageMarshaller, 187
 - ActiveMQBytesMessageMarshaller, 187
 - createObject, 187
 - getDataStructureType, 187
 - looseMarshal, 187
 - looseUnmarshal, 188
 - tightMarshal1, 188
 - tightMarshal2, 189
 - tightUnmarshal, 189
- activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller, 254
 - ~ActiveMQDestinationMarshaller, 255
 - ActiveMQDestinationMarshaller, 255
 - looseMarshal, 255
 - looseUnmarshal, 255
 - tightMarshal1, 256
 - tightMarshal2, 256
 - tightUnmarshal, 257
- activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller, 288
 - ~ActiveMQMapMessageMarshaller, 290
 - ActiveMQMapMessageMarshaller, 290
 - createObject, 290
 - getDataStructureType, 290
 - looseMarshal, 290
 - looseUnmarshal, 290
 - tightMarshal1, 291
 - tightMarshal2, 291
 - tightUnmarshal, 292
- activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller, 311
 - ~ActiveMQMessageMarshaller, 312
 - ActiveMQMessageMarshaller, 312
 - createObject, 312
 - getDataStructureType, 312
 - looseMarshal, 312

- looseUnmarshal, 313
- tightMarshal1, 313
- tightMarshal2, 314
- tightUnmarshal, 314
- activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller, 351
 - ~ActiveMQObjectMessageMarshaller, 352
 - ActiveMQObjectMessageMarshaller, 352
 - createObject, 352
 - getDataStructureType, 352
 - looseMarshal, 352
 - looseUnmarshal, 353
 - tightMarshal1, 353
 - tightMarshal2, 354
 - tightUnmarshal, 354
- activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMessageMarshaller, 386
 - ~ActiveMQQueueMarshaller, 388
 - ActiveMQQueueMarshaller, 388
 - createObject, 388
 - getDataStructureType, 388
 - looseMarshal, 388
 - looseUnmarshal, 388
 - tightMarshal1, 389
 - tightMarshal2, 389
 - tightUnmarshal, 390
- activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller, 439
 - ~ActiveMQStreamMessageMarshaller, 441
 - ActiveMQStreamMessageMarshaller, 441
 - createObject, 441
 - getDataStructureType, 441
 - looseMarshal, 441
 - looseUnmarshal, 441
 - tightMarshal1, 442
 - tightMarshal2, 442
 - tightUnmarshal, 443
- activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller, 462
 - ~ActiveMQTempDestinationMarshaller, 463
 - ActiveMQTempDestinationMarshaller, 463
 - looseMarshal, 463
 - looseUnmarshal, 464
 - tightMarshal1, 464
 - tightMarshal2, 465
 - tightUnmarshal, 465
- activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller, 485
 - ~ActiveMQTempQueueMarshaller, 486
 - ActiveMQTempQueueMarshaller, 486
 - createObject, 486
 - getDataStructureType, 486
 - looseMarshal, 486
- looseUnmarshal, 487
- tightMarshal1, 487
- tightMarshal2, 488
- tightUnmarshal, 488
- activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller, 509
 - ~ActiveMQTempTopicMarshaller, 510
 - ActiveMQTempTopicMarshaller, 510
 - createObject, 510
 - getDataStructureType, 510
 - looseMarshal, 510
 - looseUnmarshal, 511
 - tightMarshal1, 511
 - tightMarshal2, 512
 - tightUnmarshal, 512
- activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller, 533
 - ~ActiveMQTextMessageMarshaller, 535
 - ActiveMQTextMessageMarshaller, 535
 - createObject, 535
 - getDataStructureType, 535
 - looseMarshal, 535
 - looseUnmarshal, 535
 - tightMarshal1, 536
 - tightMarshal2, 536
 - tightUnmarshal, 537
- activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller, 557
 - ~ActiveMQTopicMarshaller, 558
 - ActiveMQTopicMarshaller, 558
 - createObject, 558
 - getDataStructureType, 558
 - looseMarshal, 559
 - looseUnmarshal, 559
 - tightMarshal1, 559
 - tightMarshal2, 560
 - tightUnmarshal, 560
- activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller, 610
 - ~BaseCommandMarshaller, 611
 - BaseCommandMarshaller, 611
 - looseMarshal, 611
 - looseUnmarshal, 612
 - tightMarshal1, 613
 - tightMarshal2, 614
 - tightUnmarshal, 615
- activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller, 698
 - ~BrokerIdMarshaller, 699
 - BrokerIdMarshaller, 699
 - createObject, 699
 - getDataStructureType, 699
 - looseMarshal, 699
 - looseUnmarshal, 700

- tightMarshal1, 700
- tightMarshal2, 700
- tightUnmarshal, 701
- activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller, 725
 - ~BrokerInfoMarshaller, 727
 - BrokerInfoMarshaller, 727
 - createObject, 727
 - getDataStructureType, 727
 - looseMarshal, 727
 - looseUnmarshal, 727
 - tightMarshal1, 728
 - tightMarshal2, 728
 - tightUnmarshal, 729
- activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller, 1063
 - ~ConnectionControlMarshaller, 1065
 - ConnectionControlMarshaller, 1065
 - createObject, 1065
 - getDataStructureType, 1065
 - looseMarshal, 1065
 - looseUnmarshal, 1065
 - tightMarshal1, 1066
 - tightMarshal2, 1066
 - tightUnmarshal, 1067
- activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller, 1087
 - ~ConnectionErrorMarshaller, 1088
 - ConnectionErrorMarshaller, 1088
 - createObject, 1088
 - getDataStructureType, 1088
 - looseMarshal, 1089
 - looseUnmarshal, 1089
 - tightMarshal1, 1089
 - tightMarshal2, 1090
 - tightUnmarshal, 1090
- activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller, 1113
 - ~ConnectionIdMarshaller, 1114
 - ConnectionIdMarshaller, 1114
 - createObject, 1114
 - getDataStructureType, 1114
 - looseMarshal, 1114
 - looseUnmarshal, 1115
 - tightMarshal1, 1115
 - tightMarshal2, 1115
 - tightUnmarshal, 1116
- activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller, 1142
 - ~ConnectionInfoMarshaller, 1144
 - ConnectionInfoMarshaller, 1144
 - createObject, 1144
 - getDataStructureType, 1144
 - looseMarshal, 1144
 - looseUnmarshal, 1144
 - tightMarshal1, 1145
 - tightMarshal2, 1145
 - tightUnmarshal, 1146
- activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller, 1178
 - ~ConsumerControlMarshaller, 1180
 - ConsumerControlMarshaller, 1180
 - createObject, 1180
 - getDataStructureType, 1180
 - looseMarshal, 1180
 - looseUnmarshal, 1180
 - tightMarshal1, 1181
 - tightMarshal2, 1181
- activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller, 1202
 - ~ConsumerIdMarshaller, 1204
 - ConsumerIdMarshaller, 1204
 - createObject, 1204
 - getDataStructureType, 1204
 - looseMarshal, 1204
 - looseUnmarshal, 1204
 - tightMarshal1, 1205
 - tightMarshal2, 1205
- activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller, 1226
 - ~ConsumerInfoMarshaller, 1228
 - ConsumerInfoMarshaller, 1228
 - createObject, 1228
 - getDataStructureType, 1228
 - looseMarshal, 1228
 - looseUnmarshal, 1228
 - tightMarshal1, 1229
 - tightMarshal2, 1229
- activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller, 1254
 - ~ControlCommandMarshaller, 1255
 - ControlCommandMarshaller, 1255
 - createObject, 1255
 - getDataStructureType, 1255
 - looseMarshal, 1256
 - looseUnmarshal, 1256
 - tightMarshal1, 1256
 - tightMarshal2, 1257
- activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller, 1279
 - ~DataArrayResponseMarshaller, 1280
 - createObject, 1280
 - DataArrayResponseMarshaller, 1280
 - getDataStructureType, 1280

- looseMarshal, 1280
- looseUnmarshal, 1281
- tightMarshal1, 1281
- tightMarshal2, 1282
- tightUnmarshal, 1282
- activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller, 1314
 - ~DataResponseMarshaller, 1315
 - createObject, 1315
 - DataResponseMarshaller, 1315
 - getDataStructureType, 1315
 - looseMarshal, 1315
 - looseUnmarshal, 1316
 - tightMarshal1, 1316
 - tightMarshal2, 1317
 - tightUnmarshal, 1317
- activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller, 1407
 - ~DestinationInfoMarshaller, 1408
 - createObject, 1408
 - DestinationInfoMarshaller, 1408
 - getDataStructureType, 1408
 - looseMarshal, 1408
 - looseUnmarshal, 1409
 - tightMarshal1, 1409
 - tightMarshal2, 1410
 - tightUnmarshal, 1410
- activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller, 1435
 - ~DiscoveryEventMarshaller, 1437
 - createObject, 1437
 - DiscoveryEventMarshaller, 1437
 - getDataStructureType, 1437
 - looseMarshal, 1437
 - looseUnmarshal, 1437
 - tightMarshal1, 1438
 - tightMarshal2, 1438
 - tightUnmarshal, 1439
- activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller, 1491
 - ~ExceptionResponseMarshaller, 1492
 - createObject, 1492
 - ExceptionResponseMarshaller, 1492
 - getDataStructureType, 1492
 - looseMarshal, 1492
 - looseUnmarshal, 1493
 - tightMarshal1, 1493
 - tightMarshal2, 1494
 - tightUnmarshal, 1494
- activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller, 1579
 - ~FlushCommandMarshaller, 1581
 - createObject, 1581
 - FlushCommandMarshaller, 1581
 - getDataStructureType, 1581
 - looseMarshal, 1581
 - looseUnmarshal, 1581
 - tightMarshal1, 1582
 - tightMarshal2, 1582
 - tightUnmarshal, 1582
- activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller, 1673
 - ~IntegerResponseMarshaller, 1675
 - createObject, 1675
 - getDataStructureType, 1675
 - IntegerResponseMarshaller, 1675
 - looseMarshal, 1675
 - looseUnmarshal, 1676
 - tightMarshal1, 1676
 - tightMarshal2, 1676
- activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller, 1733
 - ~JournalQueueAckMarshaller, 1735
 - createObject, 1735
 - getDataStructureType, 1735
 - JournalQueueAckMarshaller, 1735
 - looseMarshal, 1735
 - looseUnmarshal, 1735
 - tightMarshal1, 1736
 - tightMarshal2, 1736
- activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller, 1750
 - ~JournalTopicAckMarshaller, 1752
 - createObject, 1752
 - getDataStructureType, 1752
 - JournalTopicAckMarshaller, 1752
 - looseMarshal, 1752
 - looseUnmarshal, 1752
 - tightMarshal1, 1753
 - tightMarshal2, 1753
- activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller, 1781
 - ~JournalTraceMarshaller, 1783
 - createObject, 1783
 - getDataStructureType, 1783
 - JournalTraceMarshaller, 1783
 - looseMarshal, 1783
 - looseUnmarshal, 1783
 - tightMarshal1, 1784
 - tightMarshal2, 1784
- activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller, 1804
 - ~JournalTransactionMarshaller, 1806
 - createObject, 1806

- getDataSetType, 1806
- JournalTransactionMarshaller, 1806
- looseMarshal, 1806
- looseUnmarshal, 1806
- tightMarshal1, 1807
- tightMarshal2, 1807
- tightUnmarshal, 1808
- activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller, 1831
 - ~KeepAliveInfoMarshaller, 1833
 - createObject, 1833
 - getDataSetType, 1833
 - KeepAliveInfoMarshaller, 1833
 - looseMarshal, 1833
 - looseUnmarshal, 1833
 - tightMarshal1, 1834
 - tightMarshal2, 1834
 - tightUnmarshal, 1835
- activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller, 1846
 - ~LastPartialCommandMarshaller, 1848
 - createObject, 1848
 - getDataSetType, 1848
 - LastPartialCommandMarshaller, 1848
 - looseMarshal, 1848
 - looseUnmarshal, 1849
 - tightMarshal1, 1849
 - tightMarshal2, 1849
 - tightUnmarshal, 1850
- activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller, 1890
 - ~LocalTransactionIdMarshaller, 1891
 - createObject, 1891
 - getDataSetType, 1891
 - LocalTransactionIdMarshaller, 1891
 - looseMarshal, 1891
 - looseUnmarshal, 1892
 - tightMarshal1, 1892
 - tightMarshal2, 1893
 - tightUnmarshal, 1893
- activemq::wireformat::openwire::marshal::v1::MarshallerFactory, 1997
 - ~MarshallerFactory, 1997
 - configure, 1997
- activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller, 2076
 - ~MessageAckMarshaller, 2077
 - createObject, 2077
 - getDataSetType, 2077
 - looseMarshal, 2077
 - looseUnmarshal, 2078
 - MessageAckMarshaller, 2077
 - tightMarshal1, 2078
 - tightMarshal2, 2079
- tightUnmarshal, 2079
- activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller, 2107
 - ~MessageDispatchMarshaller, 2109
 - createObject, 2109
 - getDataSetType, 2109
 - looseMarshal, 2109
 - looseUnmarshal, 2109
 - MessageDispatchMarshaller, 2109
 - tightMarshal1, 2110
 - tightMarshal2, 2110
 - tightUnmarshal, 2111
- activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller, 2132
 - ~MessageDispatchNotificationMarshaller, 2133
 - createObject, 2133
 - getDataSetType, 2133
 - looseMarshal, 2133
 - looseUnmarshal, 2134
 - MessageDispatchNotificationMarshaller, 2133
 - tightMarshal1, 2134
 - tightMarshal2, 2135
 - tightUnmarshal, 2135
- activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller, 2161
 - ~MessageIdMarshaller, 2163
 - createObject, 2163
 - getDataSetType, 2163
 - looseMarshal, 2163
 - looseUnmarshal, 2163
 - MessageIdMarshaller, 2163
 - tightMarshal1, 2164
 - tightMarshal2, 2164
 - tightUnmarshal, 2165
- activemq::wireformat::openwire::marshal::v1::MessageMarshaller, 2183
 - ~MessageMarshaller, 2184
 - looseMarshal, 2184
 - looseUnmarshal, 2184
 - MessageMarshaller, 2184
 - tightMarshal1, 2185
 - tightMarshal2, 2185
 - tightUnmarshal, 2186
- activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller, 2215
 - ~MessagePullMarshaller, 2216
 - createObject, 2216
 - getDataSetType, 2216
 - looseMarshal, 2216
 - looseUnmarshal, 2217
 - MessagePullMarshaller, 2216
 - tightMarshal1, 2217

- tightMarshal2, 2218
- tightUnmarshal, 2218
- activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller, 2261
 - ~NetworkBridgeFilterMarshaller, 2263
 - createObject, 2263
 - getDataStructureType, 2263
 - looseMarshal, 2263
 - looseUnmarshal, 2263
 - NetworkBridgeFilterMarshaller, 2263
 - tightMarshal1, 2264
 - tightMarshal2, 2264
 - tightUnmarshal, 2265
- activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller, 2335
 - ~PartialCommandMarshaller, 2336
 - createObject, 2336
 - getDataStructureType, 2336
 - looseMarshal, 2336
 - looseUnmarshal, 2337
 - PartialCommandMarshaller, 2336
 - tightMarshal1, 2337
 - tightMarshal2, 2338
 - tightUnmarshal, 2338
- activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller, 2439
 - ~ProducerAckMarshaller, 2441
 - createObject, 2441
 - getDataStructureType, 2441
 - looseMarshal, 2441
 - looseUnmarshal, 2441
 - ProducerAckMarshaller, 2441
 - tightMarshal1, 2442
 - tightMarshal2, 2442
 - tightUnmarshal, 2443
- activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller, 2461
 - ~ProducerIdMarshaller, 2463
 - createObject, 2463
 - getDataStructureType, 2463
 - looseMarshal, 2463
 - looseUnmarshal, 2463
 - ProducerIdMarshaller, 2463
 - tightMarshal1, 2464
 - tightMarshal2, 2464
 - tightUnmarshal, 2465
- activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller, 2490
 - ~ProducerInfoMarshaller, 2491
 - createObject, 2491
 - getDataStructureType, 2491
 - looseMarshal, 2491
 - looseUnmarshal, 2492
 - ProducerInfoMarshaller, 2491
- tightMarshal1, 2492
- tightMarshal2, 2493
- activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller, 2540
 - ~RemoveInfoMarshaller, 2541
 - createObject, 2541
 - getDataStructureType, 2542
 - looseMarshal, 2542
 - looseUnmarshal, 2542
 - RemoveInfoMarshaller, 2541
 - tightMarshal1, 2542
 - tightMarshal2, 2543
- activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller, 2572
 - ~RemoveSubscriptionInfoMarshaller, 2573
 - createObject, 2573
 - getDataStructureType, 2573
 - looseMarshal, 2573
 - looseUnmarshal, 2574
 - RemoveSubscriptionInfoMarshaller, 2573
 - tightMarshal1, 2574
 - tightMarshal2, 2575
- activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller, 2603
 - ~ReplayCommandMarshaller, 2604
 - createObject, 2604
 - getDataStructureType, 2604
 - looseMarshal, 2604
 - looseUnmarshal, 2605
 - ReplayCommandMarshaller, 2604
 - tightMarshal1, 2605
 - tightMarshal2, 2606
- activemq::wireformat::openwire::marshal::v1::ResponseMarshaller, 2633
 - ~ResponseMarshaller, 2634
 - createObject, 2634
 - getDataStructureType, 2634
 - looseMarshal, 2635
 - looseUnmarshal, 2635
 - ResponseMarshaller, 2634
 - tightMarshal1, 2636
 - tightMarshal2, 2636
- activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller, 2685
 - ~SessionIdMarshaller, 2687
 - createObject, 2687
 - getDataStructureType, 2687
 - looseMarshal, 2687
 - looseUnmarshal, 2687

- SessionIdMarshaller, 2687
- tightMarshal1, 2688
- tightMarshal2, 2688
- tightUnmarshal, 2689
- activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller, 2709
 - ~SessionInfoMarshaller, 2710
 - createObject, 2710
 - getDataSetType, 2710
 - looseMarshal, 2710
 - looseUnmarshal, 2711
 - SessionInfoMarshaller, 2710
 - tightMarshal1, 2711
 - tightMarshal2, 2712
 - tightUnmarshal, 2712
- activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller, 2760
 - ~ShutdownInfoMarshaller, 2761
 - createObject, 2761
 - getDataSetType, 2761
 - looseMarshal, 2761
 - looseUnmarshal, 2761
 - ShutdownInfoMarshaller, 2761
 - tightMarshal1, 2762
 - tightMarshal2, 2762
 - tightUnmarshal, 2763
- activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller, 2911
 - ~SubscriptionInfoMarshaller, 2912
 - createObject, 2912
 - getDataSetType, 2913
 - looseMarshal, 2913
 - looseUnmarshal, 2913
 - SubscriptionInfoMarshaller, 2912
 - tightMarshal1, 2914
 - tightMarshal2, 2914
 - tightUnmarshal, 2914
- activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller, 3022
 - ~TransactionIdMarshaller, 3023
 - looseMarshal, 3023
 - looseUnmarshal, 3024
 - tightMarshal1, 3024
 - tightMarshal2, 3025
 - tightUnmarshal, 3025
 - TransactionIdMarshaller, 3023
- activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller, 3048
 - ~TransactionInfoMarshaller, 3050
 - createObject, 3050
 - getDataSetType, 3050
 - looseMarshal, 3050
 - looseUnmarshal, 3050
 - tightMarshal1, 3051
- tightMarshal2, 3051
- tightUnmarshal, 3052
- TransactionInfoMarshaller, 3050
- activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller, 3170
 - ~WireFormatInfoMarshaller, 3172
 - createObject, 3172
 - getDataSetType, 3172
 - looseMarshal, 3172
 - looseUnmarshal, 3172
 - tightMarshal1, 3173
 - tightMarshal2, 3173
 - tightUnmarshal, 3174
 - WireFormatInfoMarshaller, 3172
- activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller, 3209
 - ~XATransactionIdMarshaller, 3210
 - createObject, 3210
 - getDataSetType, 3210
 - looseMarshal, 3210
 - looseUnmarshal, 3211
 - tightMarshal1, 3211
 - tightMarshal2, 3212
 - tightUnmarshal, 3212
 - XATransactionIdMarshaller, 3210
- activemq::wireformat::openwire::marshal::v2, 162
 - ~ActiveMQBlobMessageMarshaller, 164
 - ActiveMQBlobMessageMarshaller, 164
 - createObject, 164
 - getDataSetType, 164
 - looseMarshal, 164
 - looseUnmarshal, 164
 - tightMarshal1, 165
 - tightMarshal2, 165
- activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller, 162
 - ~ActiveMQBlobMessageMarshaller, 164
 - ActiveMQBlobMessageMarshaller, 164
 - createObject, 164
 - getDataSetType, 164
 - looseMarshal, 164
 - looseUnmarshal, 164
 - tightMarshal1, 165
 - tightMarshal2, 165
- activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller, 198
 - ~ActiveMQBytesMessageMarshaller, 199
 - ActiveMQBytesMessageMarshaller, 199
 - createObject, 199
 - getDataSetType, 199
 - looseMarshal, 199
 - looseUnmarshal, 200
 - tightMarshal1, 200
 - tightMarshal2, 201
 - tightUnmarshal, 201
- activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller, 265
 - ~ActiveMQDestinationMarshaller, 267
 - ActiveMQDestinationMarshaller, 267
 - looseMarshal, 267

- looseUnmarshal, 267
- tightMarshal1, 268
- tightMarshal2, 268
- tightUnmarshal, 269
- activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller, 300
 - ~ActiveMQMapMessageMarshaller, 302
 - ActiveMQMapMessageMarshaller, 302
 - createObject, 302
 - getDataStructureType, 302
 - looseMarshal, 302
 - looseUnmarshal, 302
 - tightMarshal1, 303
 - tightMarshal2, 303
 - tightUnmarshal, 304
- activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller, 323
 - ~ActiveMQMessageMarshaller, 324
 - ActiveMQMessageMarshaller, 324
 - createObject, 324
 - getDataStructureType, 324
 - looseMarshal, 324
 - looseUnmarshal, 325
 - tightMarshal1, 325
 - tightMarshal2, 326
 - tightUnmarshal, 326
- activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller, 363
 - ~ActiveMQObjectMessageMarshaller, 364
 - ActiveMQObjectMessageMarshaller, 364
 - createObject, 364
 - getDataStructureType, 364
 - looseMarshal, 364
 - looseUnmarshal, 365
 - tightMarshal1, 365
 - tightMarshal2, 366
 - tightUnmarshal, 366
- activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller, 398
 - ~ActiveMQQueueMarshaller, 400
 - ActiveMQQueueMarshaller, 400
 - createObject, 400
 - getDataStructureType, 400
 - looseMarshal, 400
 - looseUnmarshal, 400
 - tightMarshal1, 401
 - tightMarshal2, 401
 - tightUnmarshal, 402
- activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller, 451
 - ~ActiveMQStreamMessageMarshaller, 453
 - ActiveMQStreamMessageMarshaller, 453
 - createObject, 453
 - getDataStructureType, 453
- looseMarshal, 453
- looseUnmarshal, 453
- tightMarshal1, 454
- tightMarshal2, 454
- activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller, 473
 - ~ActiveMQTempDestinationMarshaller, 474
 - ActiveMQTempDestinationMarshaller, 474
 - looseMarshal, 474
 - looseUnmarshal, 475
 - tightMarshal1, 475
 - tightMarshal2, 476
 - tightUnmarshal, 476
- activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller, 497
 - ~ActiveMQTempQueueMarshaller, 498
 - ActiveMQTempQueueMarshaller, 498
 - createObject, 498
 - getDataStructureType, 498
 - looseMarshal, 498
 - looseUnmarshal, 499
 - tightMarshal1, 499
 - tightMarshal2, 500
 - tightUnmarshal, 500
- activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller, 521
 - ~ActiveMQTempTopicMarshaller, 522
 - ActiveMQTempTopicMarshaller, 522
 - createObject, 522
 - getDataStructureType, 522
 - looseMarshal, 522
 - looseUnmarshal, 523
 - tightMarshal1, 523
 - tightMarshal2, 524
 - tightUnmarshal, 524
- activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller, 545
 - ~ActiveMQTextMessageMarshaller, 547
 - ActiveMQTextMessageMarshaller, 547
 - createObject, 547
 - getDataStructureType, 547
 - looseMarshal, 547
 - looseUnmarshal, 547
 - tightMarshal1, 548
 - tightMarshal2, 548
 - tightUnmarshal, 549
- activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller, 569
 - ~ActiveMQTopicMarshaller, 570
 - ActiveMQTopicMarshaller, 570
 - createObject, 570
 - getDataStructureType, 570

- looseMarshal, 570
 - looseUnmarshal, 571
 - tightMarshal1, 571
 - tightMarshal2, 572
 - tightUnmarshal, 572
- activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller, 630
 - ~BaseCommandMarshaller, 631
 - BaseCommandMarshaller, 631
 - looseMarshal, 631
 - looseUnmarshal, 632
 - tightMarshal1, 633
 - tightMarshal2, 634
 - tightUnmarshal, 635
- activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller, 709
 - ~BrokerIdMarshaller, 711
 - BrokerIdMarshaller, 711
 - createObject, 711
 - getDataStructureType, 711
 - looseMarshal, 711
 - looseUnmarshal, 711
 - tightMarshal1, 712
 - tightMarshal2, 712
 - tightUnmarshal, 713
- activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller, 737
 - ~BrokerInfoMarshaller, 739
 - BrokerInfoMarshaller, 739
 - createObject, 739
 - getDataStructureType, 739
 - looseMarshal, 739
 - looseUnmarshal, 739
 - tightMarshal1, 740
 - tightMarshal2, 740
 - tightUnmarshal, 741
- activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller, 1075
 - ~ConnectionControlMarshaller, 1077
 - ConnectionControlMarshaller, 1077
 - createObject, 1077
 - getDataStructureType, 1077
 - looseMarshal, 1077
 - looseUnmarshal, 1077
 - tightMarshal1, 1078
 - tightMarshal2, 1078
 - tightUnmarshal, 1079
- activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller, 1099
 - ~ConnectionErrorMarshaller, 1100
 - ConnectionErrorMarshaller, 1100
 - createObject, 1100
 - getDataStructureType, 1100
 - looseMarshal, 1100
- looseUnmarshal, 1101
- tightMarshal1, 1101
- tightMarshal2, 1102
- tightUnmarshal, 1102
- activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller, 1124
 - ~ConnectionIdMarshaller, 1126
 - ConnectionIdMarshaller, 1126
 - createObject, 1126
 - getDataStructureType, 1126
 - looseMarshal, 1126
 - looseUnmarshal, 1126
 - tightMarshal1, 1127
 - tightMarshal2, 1127
- activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller, 1150
 - ~ConnectionInfoMarshaller, 1152
 - ConnectionInfoMarshaller, 1152
 - createObject, 1152
 - getDataStructureType, 1152
 - looseMarshal, 1152
 - looseUnmarshal, 1152
 - tightMarshal1, 1153
 - tightMarshal2, 1153
- activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller, 1186
 - ~ConsumerControlMarshaller, 1188
 - ConsumerControlMarshaller, 1188
 - createObject, 1188
 - getDataStructureType, 1188
 - looseMarshal, 1188
 - looseUnmarshal, 1188
 - tightMarshal1, 1189
 - tightMarshal2, 1189
- activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller, 1210
 - ~ConsumerIdMarshaller, 1212
 - ConsumerIdMarshaller, 1212
 - createObject, 1212
 - getDataStructureType, 1212
 - looseMarshal, 1212
 - looseUnmarshal, 1212
 - tightMarshal1, 1213
 - tightMarshal2, 1213
- activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller, 1238
 - ~ConsumerInfoMarshaller, 1240
 - ConsumerInfoMarshaller, 1240
 - createObject, 1240
 - getDataStructureType, 1240

- looseMarshal, 1240
- looseUnmarshal, 1240
- tightMarshal1, 1241
- tightMarshal2, 1241
- tightUnmarshal, 1242
- activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller, 1262
 - ~ControlCommandMarshaller, 1263
 - ControlCommandMarshaller, 1263
 - createObject, 1263
 - getDataStructureType, 1263
 - looseMarshal, 1263
 - looseUnmarshal, 1264
 - tightMarshal1, 1264
 - tightMarshal2, 1265
 - tightUnmarshal, 1265
- activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller, 1287
 - ~DataArrayResponseMarshaller, 1288
 - createObject, 1288
 - DataArrayResponseMarshaller, 1288
 - getDataStructureType, 1288
 - looseMarshal, 1288
 - looseUnmarshal, 1289
 - tightMarshal1, 1289
 - tightMarshal2, 1290
 - tightUnmarshal, 1290
- activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller, 1326
 - ~DataResponseMarshaller, 1327
 - createObject, 1327
 - DataResponseMarshaller, 1327
 - getDataStructureType, 1327
 - looseMarshal, 1327
 - looseUnmarshal, 1328
 - tightMarshal1, 1328
 - tightMarshal2, 1329
 - tightUnmarshal, 1329
- activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller, 1395
 - ~DestinationInfoMarshaller, 1396
 - createObject, 1396
 - DestinationInfoMarshaller, 1396
 - getDataStructureType, 1397
 - looseMarshal, 1397
 - looseUnmarshal, 1397
 - tightMarshal1, 1398
 - tightMarshal2, 1398
 - tightUnmarshal, 1398
- activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller, 1420
 - ~DiscoveryEventMarshaller, 1421
 - createObject, 1421
 - DiscoveryEventMarshaller, 1421
 - getDataStructureType, 1421
 - looseMarshal, 1421
 - looseUnmarshal, 1422
 - tightMarshal1, 1422
 - tightMarshal2, 1423
 - tightUnmarshal, 1423
- activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller, 1487
 - ~ExceptionResponseMarshaller, 1488
 - createObject, 1488
 - ExceptionResponseMarshaller, 1488
 - getDataStructureType, 1488
 - looseMarshal, 1488
 - looseUnmarshal, 1489
 - tightMarshal1, 1489
 - tightMarshal2, 1490
 - tightUnmarshal, 1490
- activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller, 1576
 - ~FlushCommandMarshaller, 1577
 - createObject, 1577
 - FlushCommandMarshaller, 1577
 - getDataStructureType, 1577
 - looseMarshal, 1577
 - looseUnmarshal, 1578
 - tightMarshal1, 1578
 - tightMarshal2, 1578
- activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller, 1669
 - ~IntegerResponseMarshaller, 1671
 - createObject, 1671
 - getDataStructureType, 1671
 - IntegerResponseMarshaller, 1671
 - looseMarshal, 1671
 - looseUnmarshal, 1672
 - tightMarshal1, 1672
 - tightMarshal2, 1672
- activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller, 1722
 - ~JournalQueueAckMarshaller, 1723
 - createObject, 1723
 - getDataStructureType, 1723
 - JournalQueueAckMarshaller, 1723
 - looseMarshal, 1723
 - looseUnmarshal, 1724
 - tightMarshal1, 1724
 - tightMarshal2, 1724
- activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller, 1746
 - ~JournalTopicAckMarshaller, 1748
 - createObject, 1748

- getDataStructureType, 1748
 - JournalTopicAckMarshaller, 1748
 - looseMarshal, 1748
 - looseUnmarshal, 1748
 - tightMarshal1, 1749
 - tightMarshal2, 1749
 - tightUnmarshal, 1750
- activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller, 1769
 - ~JournalTraceMarshaller, 1771
 - createObject, 1771
 - getDataStructureType, 1771
 - JournalTraceMarshaller, 1771
 - looseMarshal, 1771
 - looseUnmarshal, 1771
 - tightMarshal1, 1772
 - tightMarshal2, 1772
 - tightUnmarshal, 1773
- activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller, 1793
 - ~JournalTransactionMarshaller, 1794
 - createObject, 1794
 - getDataStructureType, 1794
 - JournalTransactionMarshaller, 1794
 - looseMarshal, 1795
 - looseUnmarshal, 1795
 - tightMarshal1, 1795
 - tightMarshal2, 1796
 - tightUnmarshal, 1796
- activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller, 1815
 - ~KeepAliveInfoMarshaller, 1817
 - createObject, 1817
 - getDataStructureType, 1817
 - KeepAliveInfoMarshaller, 1817
 - looseMarshal, 1817
 - looseUnmarshal, 1817
 - tightMarshal1, 1818
 - tightMarshal2, 1818
 - tightUnmarshal, 1819
- activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller, 1842
 - ~LastPartialCommandMarshaller, 1844
 - createObject, 1844
 - getDataStructureType, 1844
 - LastPartialCommandMarshaller, 1844
 - looseMarshal, 1844
 - looseUnmarshal, 1845
 - tightMarshal1, 1845
 - tightMarshal2, 1845
 - tightUnmarshal, 1846
- activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller, 1878
 - ~LocalTransactionIdMarshaller, 1879
 - createObject, 1879
 - getDataStructureType, 1879
 - LocalTransactionIdMarshaller, 1879
 - looseMarshal, 1879
 - looseUnmarshal, 1880
 - tightMarshal1, 1880
 - tightMarshal2, 1881
 - tightUnmarshal, 1881
- activemq::wireformat::openwire::marshal::v2::MarshallerFactory, 1995
 - ~MarshallerFactory, 1996
 - configure, 1996
- activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller, 2060
 - ~MessageAckMarshaller, 2061
 - createObject, 2061
 - getDataStructureType, 2062
 - looseMarshal, 2062
 - looseUnmarshal, 2062
 - MessageAckMarshaller, 2061
 - tightMarshal1, 2062
 - tightMarshal2, 2063
 - tightUnmarshal, 2063
- activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller, 2095
 - ~MessageDispatchMarshaller, 2097
 - createObject, 2097
 - getDataStructureType, 2097
 - looseMarshal, 2097
 - looseUnmarshal, 2097
 - MessageDispatchMarshaller, 2097
 - tightMarshal1, 2098
 - tightMarshal2, 2098
 - tightUnmarshal, 2099
- activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller, 2120
 - ~MessageDispatchNotificationMarshaller, 2121
 - createObject, 2121
 - getDataStructureType, 2121
 - looseMarshal, 2122
 - looseUnmarshal, 2122
 - MessageDispatchNotificationMarshaller, 2121
 - tightMarshal1, 2122
 - tightMarshal2, 2123
 - tightUnmarshal, 2123
- activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller, 2146
 - ~MessageIdMarshaller, 2147
 - createObject, 2147
 - getDataStructureType, 2147
 - looseMarshal, 2148
 - looseUnmarshal, 2148

- MessageIdMarshaller, 2147
- tightMarshal1, 2148
- tightMarshal2, 2149
- tightUnmarshal, 2149
- activemq::wireformat::openwire::marshal::v2::MessageMarshaller, 2166
 - ~MessageMarshaller, 2167
 - looseMarshal, 2167
 - looseUnmarshal, 2168
 - MessageMarshaller, 2167
 - tightMarshal1, 2168
 - tightMarshal2, 2169
 - tightUnmarshal, 2170
- activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller, 2207
 - ~MessagePullMarshaller, 2208
 - createObject, 2208
 - getDataStructureType, 2209
 - looseMarshal, 2209
 - looseUnmarshal, 2209
 - MessagePullMarshaller, 2208
 - tightMarshal1, 2209
 - tightMarshal2, 2210
 - tightUnmarshal, 2210
- activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller, 2250
 - ~NetworkBridgeFilterMarshaller, 2251
 - createObject, 2251
 - getDataStructureType, 2251
 - looseMarshal, 2251
 - looseUnmarshal, 2252
 - NetworkBridgeFilterMarshaller, 2251
 - tightMarshal1, 2252
 - tightMarshal2, 2252
 - tightUnmarshal, 2253
- activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller, 2322
 - ~PartialCommandMarshaller, 2323
 - createObject, 2323
 - getDataStructureType, 2323
 - looseMarshal, 2324
 - looseUnmarshal, 2324
 - PartialCommandMarshaller, 2323
 - tightMarshal1, 2324
 - tightMarshal2, 2325
 - tightUnmarshal, 2325
- activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller, 2424
 - ~ProducerAckMarshaller, 2425
 - createObject, 2425
 - getDataStructureType, 2425
 - looseMarshal, 2425
 - looseUnmarshal, 2425
 - ProducerAckMarshaller, 2425
- tightMarshal1, 2426
- tightMarshal2, 2426
- tightUnmarshal, 2427
- activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller, 2450
 - ~ProducerIdMarshaller, 2451
 - createObject, 2451
 - getDataStructureType, 2451
 - looseMarshal, 2451
 - looseUnmarshal, 2452
 - ProducerIdMarshaller, 2451
 - tightMarshal1, 2452
 - tightMarshal2, 2453
- activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller, 2474
 - ~ProducerInfoMarshaller, 2475
 - createObject, 2475
 - getDataStructureType, 2475
 - looseMarshal, 2476
 - looseUnmarshal, 2476
 - ProducerInfoMarshaller, 2475
 - tightMarshal1, 2476
 - tightMarshal2, 2477
- activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller, 2548
 - ~RemoveInfoMarshaller, 2549
 - createObject, 2549
 - getDataStructureType, 2549
 - looseMarshal, 2549
 - looseUnmarshal, 2550
 - RemoveInfoMarshaller, 2549
 - tightMarshal1, 2550
 - tightMarshal2, 2551
- activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller, 2568
 - ~RemoveSubscriptionInfoMarshaller, 2569
 - createObject, 2569
 - getDataStructureType, 2569
 - looseMarshal, 2570
 - looseUnmarshal, 2570
 - RemoveSubscriptionInfoMarshaller, 2569
 - tightMarshal1, 2570
 - tightMarshal2, 2571
- activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller, 2587
 - ~ReplayCommandMarshaller, 2588
 - createObject, 2588
 - getDataStructureType, 2589
 - looseMarshal, 2589
 - looseUnmarshal, 2589

- ReplayCommandMarshaller, 2588
- tightMarshal1, 2589
- tightMarshal2, 2590
- tightUnmarshal, 2590
- activemq::wireformat::openwire::marshal::v2::ResponseMarshaller,
 - 2620
 - ~ResponseMarshaller, 2621
 - createObject, 2621
 - getDataStructureType, 2621
 - looseMarshal, 2621
 - looseUnmarshal, 2622
 - ResponseMarshaller, 2621
 - tightMarshal1, 2622
 - tightMarshal2, 2623
 - tightUnmarshal, 2623
- activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller,
 - 2689
 - ~SessionIdMarshaller, 2691
 - createObject, 2691
 - getDataStructureType, 2691
 - looseMarshal, 2691
 - looseUnmarshal, 2691
 - SessionIdMarshaller, 2691
 - tightMarshal1, 2692
 - tightMarshal2, 2692
 - tightUnmarshal, 2693
- activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller,
 - 2705
 - ~SessionInfoMarshaller, 2706
 - createObject, 2706
 - getDataStructureType, 2706
 - looseMarshal, 2707
 - looseUnmarshal, 2707
 - SessionInfoMarshaller, 2706
 - tightMarshal1, 2707
 - tightMarshal2, 2708
 - tightUnmarshal, 2708
- activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller,
 - 2771
 - ~ShutdownInfoMarshaller, 2773
 - createObject, 2773
 - getDataStructureType, 2773
 - looseMarshal, 2773
 - looseUnmarshal, 2773
 - ShutdownInfoMarshaller, 2773
 - tightMarshal1, 2774
 - tightMarshal2, 2774
 - tightUnmarshal, 2775
- activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller,
 - 2926
 - ~SubscriptionInfoMarshaller, 2928
 - createObject, 2928
 - getDataStructureType, 2928
 - looseMarshal, 2928
- looseUnmarshal, 2928
- SubscriptionInfoMarshaller, 2928
- tightMarshal1, 2929
- tightMarshal2, 2929
- TransactionIdMarshaller, 2930
- activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller,
 - 3018
 - ~TransactionIdMarshaller, 3020
 - looseMarshal, 3020
 - looseUnmarshal, 3020
 - tightMarshal1, 3021
 - tightMarshal2, 3021
 - tightUnmarshal, 3022
 - TransactionIdMarshaller, 3020
- activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller,
 - 3056
 - ~TransactionInfoMarshaller, 3058
 - createObject, 3058
 - getDataStructureType, 3058
 - looseMarshal, 3058
 - looseUnmarshal, 3058
 - tightMarshal1, 3059
 - tightMarshal2, 3059
 - tightUnmarshal, 3060
 - TransactionInfoMarshaller, 3058
- activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller,
 - 3163
 - ~WireFormatInfoMarshaller, 3164
 - createObject, 3164
 - getDataStructureType, 3164
 - looseMarshal, 3164
 - looseUnmarshal, 3165
 - tightMarshal1, 3165
 - tightMarshal2, 3165
 - tightUnmarshal, 3166
 - WireFormatInfoMarshaller, 3164
- activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller,
 - 3197
 - ~XATransactionIdMarshaller, 3198
 - createObject, 3198
 - getDataStructureType, 3198
 - looseMarshal, 3199
 - looseUnmarshal, 3199
 - tightMarshal1, 3199
 - tightMarshal2, 3200
 - tightUnmarshal, 3200
 - XATransactionIdMarshaller, 3198
- activemq::wireformat::openwire::marshal::v3,
- activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller,
 - 146
 - ~ActiveMQBlobMessageMarshaller, 148
 - ActiveMQBlobMessageMarshaller, 148
 - createObject, 148

- getDataStructureType, 148
 - looseMarshal, 148
 - looseUnmarshal, 148
 - tightMarshal1, 149
 - tightMarshal2, 149
 - tightUnmarshal, 150
- activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller, 182
 - ~ActiveMQBytesMessageMarshaller, 183
 - ActiveMQBytesMessageMarshaller, 183
 - createObject, 183
 - getDataStructureType, 183
 - looseMarshal, 183
 - looseUnmarshal, 184
 - tightMarshal1, 184
 - tightMarshal2, 185
 - tightUnmarshal, 185
- activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller, 250
 - ~ActiveMQDestinationMarshaller, 251
 - ActiveMQDestinationMarshaller, 251
 - looseMarshal, 251
 - looseUnmarshal, 252
 - tightMarshal1, 252
 - tightMarshal2, 253
 - tightUnmarshal, 253
- activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller, 284
 - ~ActiveMQMapMessageMarshaller, 286
 - ActiveMQMapMessageMarshaller, 286
 - createObject, 286
 - getDataStructureType, 286
 - looseMarshal, 286
 - looseUnmarshal, 286
 - tightMarshal1, 287
 - tightMarshal2, 287
 - tightUnmarshal, 288
- activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller, 307
 - ~ActiveMQMessageMarshaller, 308
 - ActiveMQMessageMarshaller, 308
 - createObject, 308
 - getDataStructureType, 308
 - looseMarshal, 309
 - looseUnmarshal, 309
 - tightMarshal1, 309
 - tightMarshal2, 310
 - tightUnmarshal, 310
- activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller, 347
 - ~ActiveMQObjectMessageMarshaller, 348
 - ActiveMQObjectMessageMarshaller, 348
 - createObject, 348
 - getDataStructureType, 348
- looseMarshal, 348
- looseUnmarshal, 349
- tightMarshal1, 349
- tightMarshal2, 350
- tightUnmarshal, 350
- activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller, 384
 - ~ActiveMQQueueMarshaller, 384
 - ActiveMQQueueMarshaller, 384
 - createObject, 384
 - getDataStructureType, 384
 - looseMarshal, 384
 - looseUnmarshal, 384
 - tightMarshal1, 385
 - tightMarshal2, 385
 - tightUnmarshal, 386
- activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller, 436
 - ~ActiveMQStreamMessageMarshaller, 437
 - ActiveMQStreamMessageMarshaller, 437
 - createObject, 437
 - getDataStructureType, 437
 - looseMarshal, 437
 - looseUnmarshal, 438
 - tightMarshal1, 438
 - tightMarshal2, 438
- activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller, 459
 - ~ActiveMQTempDestinationMarshaller, 460
 - ActiveMQTempDestinationMarshaller, 460
 - looseMarshal, 460
 - looseUnmarshal, 460
 - tightMarshal1, 461
 - tightMarshal2, 461
 - tightUnmarshal, 462
- activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller, 481
 - ~ActiveMQTempQueueMarshaller, 482
 - ActiveMQTempQueueMarshaller, 482
 - createObject, 482
 - getDataStructureType, 482
 - looseMarshal, 483
 - looseUnmarshal, 483
 - tightMarshal1, 483
 - tightMarshal2, 484
 - tightUnmarshal, 484
- activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller, 505
 - ~ActiveMQTempTopicMarshaller, 506
 - ActiveMQTempTopicMarshaller, 506
 - createObject, 506
 - getDataStructureType, 506

- looseMarshal, 506
- looseUnmarshal, 507
- tightMarshal1, 507
- tightMarshal2, 508
- tightUnmarshal, 508
- activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller, 529
 - ~ActiveMQTextMessageMarshaller, 531
 - ActiveMQTextMessageMarshaller, 531
 - createObject, 531
 - getDataStructureType, 531
 - looseMarshal, 531
 - looseUnmarshal, 531
 - tightMarshal1, 532
 - tightMarshal2, 532
 - tightUnmarshal, 533
- activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller, 553
 - ~ActiveMQTopicMarshaller, 554
 - ActiveMQTopicMarshaller, 554
 - createObject, 554
 - getDataStructureType, 555
 - looseMarshal, 555
 - looseUnmarshal, 555
 - tightMarshal1, 556
 - tightMarshal2, 556
 - tightUnmarshal, 556
- activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller, 603
 - ~BaseCommandMarshaller, 604
 - BaseCommandMarshaller, 604
 - looseMarshal, 604
 - looseUnmarshal, 605
 - tightMarshal1, 606
 - tightMarshal2, 607
 - tightUnmarshal, 609
- activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller, 694
 - ~BrokerIdMarshaller, 695
 - BrokerIdMarshaller, 695
 - createObject, 695
 - getDataStructureType, 695
 - looseMarshal, 695
 - looseUnmarshal, 696
 - tightMarshal1, 696
 - tightMarshal2, 697
 - tightUnmarshal, 697
- activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller, 721
 - ~BrokerInfoMarshaller, 723
 - BrokerInfoMarshaller, 723
 - createObject, 723
 - getDataStructureType, 723
 - looseMarshal, 723
 - looseUnmarshal, 723
 - tightMarshal1, 724
 - tightMarshal2, 724
 - tightUnmarshal, 725
- activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller, 1061
 - ~ConnectionControlMarshaller, 1061
 - ConnectionControlMarshaller, 1061
 - createObject, 1061
 - getDataStructureType, 1061
 - looseMarshal, 1061
 - looseUnmarshal, 1061
 - tightMarshal1, 1062
 - tightMarshal2, 1062
 - tightUnmarshal, 1063
- activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller, 1084
 - ~ConnectionErrorMarshaller, 1084
 - ConnectionErrorMarshaller, 1084
 - createObject, 1084
 - getDataStructureType, 1085
 - looseMarshal, 1085
 - looseUnmarshal, 1085
 - tightMarshal1, 1086
 - tightMarshal2, 1086
 - tightUnmarshal, 1086
- activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller, 1110
 - ~ConnectionIdMarshaller, 1110
 - ConnectionIdMarshaller, 1110
 - createObject, 1110
 - getDataStructureType, 1110
 - looseMarshal, 1110
 - looseUnmarshal, 1111
 - tightMarshal1, 1111
 - tightMarshal2, 1112
 - tightUnmarshal, 1112
- activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller, 1134
 - ~ConnectionInfoMarshaller, 1136
 - ConnectionInfoMarshaller, 1136
 - createObject, 1136
 - getDataStructureType, 1136
 - looseMarshal, 1136
 - looseUnmarshal, 1136
 - tightMarshal1, 1137
 - tightMarshal2, 1137
 - tightUnmarshal, 1138
- activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller, 1170
 - ~ConsumerControlMarshaller, 1172
 - ConsumerControlMarshaller, 1172
 - createObject, 1172
 - getDataStructureType, 1172

- looseMarshal, 1172
- looseUnmarshal, 1172
- tightMarshal1, 1173
- tightMarshal2, 1173
- tightUnmarshal, 1174
- activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller, 1195
 - ~ConsumerIdMarshaller, 1196
 - ConsumerIdMarshaller, 1196
 - createObject, 1196
 - getDataStructureType, 1196
 - looseMarshal, 1196
 - looseUnmarshal, 1197
 - tightMarshal1, 1197
 - tightMarshal2, 1197
 - tightUnmarshal, 1198
- activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller, 1223
 - ~ConsumerInfoMarshaller, 1224
 - ConsumerInfoMarshaller, 1224
 - createObject, 1224
 - getDataStructureType, 1224
 - looseMarshal, 1224
 - looseUnmarshal, 1225
 - tightMarshal1, 1225
 - tightMarshal2, 1225
 - tightUnmarshal, 1226
- activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller, 1246
 - ~ControlCommandMarshaller, 1247
 - ControlCommandMarshaller, 1247
 - createObject, 1247
 - getDataStructureType, 1248
 - looseMarshal, 1248
 - looseUnmarshal, 1248
 - tightMarshal1, 1249
 - tightMarshal2, 1249
 - tightUnmarshal, 1249
- activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller, 1271
 - ~DataArrayResponseMarshaller, 1272
 - createObject, 1272
 - DataArrayResponseMarshaller, 1272
 - getDataStructureType, 1272
 - looseMarshal, 1272
 - looseUnmarshal, 1273
 - tightMarshal1, 1273
 - tightMarshal2, 1274
 - tightUnmarshal, 1274
- activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller, 1310
 - ~DataResponseMarshaller, 1311
 - createObject, 1311
 - DataResponseMarshaller, 1311
- getDataStructureType, 1311
- looseMarshal, 1311
- looseUnmarshal, 1312
- tightMarshal1, 1312
- tightMarshal2, 1313
- activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller, 1399
 - ~DestinationInfoMarshaller, 1400
 - createObject, 1400
 - DestinationInfoMarshaller, 1400
 - getDataStructureType, 1400
 - looseMarshal, 1401
 - looseUnmarshal, 1401
 - tightMarshal1, 1401
 - tightMarshal2, 1402
- activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller, 1424
 - ~DiscoveryEventMarshaller, 1425
 - createObject, 1425
 - DiscoveryEventMarshaller, 1425
 - getDataStructureType, 1425
 - looseMarshal, 1425
 - looseUnmarshal, 1426
 - tightMarshal1, 1426
 - tightMarshal2, 1426
- activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller, 1495
 - ~ExceptionResponseMarshaller, 1496
 - createObject, 1496
 - ExceptionResponseMarshaller, 1496
 - getDataStructureType, 1496
 - looseMarshal, 1496
 - looseUnmarshal, 1497
 - tightMarshal1, 1497
 - tightMarshal2, 1498
- activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller, 1583
 - ~FlushCommandMarshaller, 1585
 - createObject, 1585
 - FlushCommandMarshaller, 1585
 - getDataStructureType, 1585
 - looseMarshal, 1585
 - looseUnmarshal, 1585
 - tightMarshal1, 1586
 - tightMarshal2, 1586
- activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller, 1677
 - ~IntegerResponseMarshaller, 1679
 - createObject, 1679

- getDataStructureType, 1679
 - IntegerResponseMarshaller, 1679
 - looseMarshal, 1679
 - looseUnmarshal, 1680
 - tightMarshal1, 1680
 - tightMarshal2, 1680
 - tightUnmarshal, 1681
- activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller, 1729
 - ~JournalQueueAckMarshaller, 1731
 - createObject, 1731
 - getDataStructureType, 1731
 - JournalQueueAckMarshaller, 1731
 - looseMarshal, 1731
 - looseUnmarshal, 1731
 - tightMarshal1, 1732
 - tightMarshal2, 1732
 - tightUnmarshal, 1733
- activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller, 1754
 - ~JournalTopicAckMarshaller, 1756
 - createObject, 1756
 - getDataStructureType, 1756
 - JournalTopicAckMarshaller, 1756
 - looseMarshal, 1756
 - looseUnmarshal, 1756
 - tightMarshal1, 1757
 - tightMarshal2, 1757
 - tightUnmarshal, 1758
- activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller, 1777
 - ~JournalTraceMarshaller, 1779
 - createObject, 1779
 - getDataStructureType, 1779
 - JournalTraceMarshaller, 1779
 - looseMarshal, 1779
 - looseUnmarshal, 1779
 - tightMarshal1, 1780
 - tightMarshal2, 1780
 - tightUnmarshal, 1781
- activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller, 1797
 - ~JournalTransactionMarshaller, 1798
 - createObject, 1798
 - getDataStructureType, 1798
 - JournalTransactionMarshaller, 1798
 - looseMarshal, 1798
 - looseUnmarshal, 1799
 - tightMarshal1, 1799
 - tightMarshal2, 1800
 - tightUnmarshal, 1800
- activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller, 1823
 - ~KeepAliveInfoMarshaller, 1825
 - createObject, 1825
 - getDataStructureType, 1825
 - KeepAliveInfoMarshaller, 1825
 - looseMarshal, 1825
 - looseUnmarshal, 1825
 - tightMarshal1, 1826
 - tightMarshal2, 1826
- activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller, 1850
 - ~LastPartialCommandMarshaller, 1852
 - createObject, 1852
 - getDataStructureType, 1852
 - LastPartialCommandMarshaller, 1852
 - looseMarshal, 1852
 - looseUnmarshal, 1853
 - tightMarshal1, 1853
 - tightMarshal2, 1853
- activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller, 1882
 - ~LocalTransactionIdMarshaller, 1883
 - createObject, 1883
 - getDataStructureType, 1883
 - LocalTransactionIdMarshaller, 1883
 - looseMarshal, 1883
 - looseUnmarshal, 1884
 - tightMarshal1, 1884
 - tightMarshal2, 1885
- activemq::wireformat::openwire::marshal::v3::MarshallerFactory, 1997
 - ~MarshallerFactory, 1998
 - configure, 1998
- activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller, 2068
 - ~MessageAckMarshaller, 2069
 - createObject, 2069
 - getDataStructureType, 2069
 - looseMarshal, 2069
 - looseUnmarshal, 2070
 - MessageAckMarshaller, 2069
 - tightMarshal1, 2070
 - tightMarshal2, 2071
 - tightUnmarshal, 2071
- activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller, 2103
 - ~MessageDispatchMarshaller, 2105
 - createObject, 2105
 - getDataStructureType, 2105
 - looseMarshal, 2105
 - looseUnmarshal, 2105
 - MessageDispatchMarshaller, 2105
 - tightMarshal1, 2106

- tightMarshal2, 2106
- tightUnmarshal, 2107
- activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller, 2128
 - ~MessageDispatchNotificationMarshaller, 2129
 - createObject, 2129
 - getDataStructureType, 2129
 - looseMarshal, 2129
 - looseUnmarshal, 2130
 - MessageDispatchNotificationMarshaller, 2129
 - tightMarshal1, 2130
 - tightMarshal2, 2131
 - tightUnmarshal, 2131
- activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller, 2154
 - ~MessageIdMarshaller, 2155
 - createObject, 2155
 - getDataStructureType, 2155
 - looseMarshal, 2155
 - looseUnmarshal, 2156
 - MessageIdMarshaller, 2155
 - tightMarshal1, 2156
 - tightMarshal2, 2156
 - tightUnmarshal, 2157
- activemq::wireformat::openwire::marshal::v3::MessageMarshaller, 2175
 - ~MessageMarshaller, 2176
 - looseMarshal, 2176
 - looseUnmarshal, 2176
 - MessageMarshaller, 2176
 - tightMarshal1, 2177
 - tightMarshal2, 2177
 - tightUnmarshal, 2178
- activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller, 2211
 - ~MessagePullMarshaller, 2212
 - createObject, 2212
 - getDataStructureType, 2212
 - looseMarshal, 2212
 - looseUnmarshal, 2213
 - MessagePullMarshaller, 2212
 - tightMarshal1, 2213
 - tightMarshal2, 2214
 - tightUnmarshal, 2214
- activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller, 2253
 - ~NetworkBridgeFilterMarshaller, 2255
 - createObject, 2255
 - getDataStructureType, 2255
 - looseMarshal, 2255
 - looseUnmarshal, 2255
 - NetworkBridgeFilterMarshaller, 2255
- tightMarshal1, 2256
- tightMarshal2, 2256
- activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller, 2326
 - ~PartialCommandMarshaller, 2327
 - createObject, 2327
 - getDataStructureType, 2328
 - looseMarshal, 2328
 - looseUnmarshal, 2328
 - PartialCommandMarshaller, 2327
 - tightMarshal1, 2329
 - tightMarshal2, 2329
 - tightUnmarshal, 2330
- activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller, 2429
 - ~ProducerAckMarshaller, 2429
 - createObject, 2429
 - getDataStructureType, 2429
 - looseMarshal, 2429
 - looseUnmarshal, 2429
 - ProducerAckMarshaller, 2429
 - tightMarshal1, 2430
 - tightMarshal2, 2430
 - tightUnmarshal, 2431
- activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller, 2455
 - ~ProducerIdMarshaller, 2455
 - createObject, 2455
 - getDataStructureType, 2455
 - looseMarshal, 2455
 - looseUnmarshal, 2456
 - ProducerIdMarshaller, 2455
 - tightMarshal1, 2456
 - tightMarshal2, 2456
- activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller, 2478
 - ~ProducerInfoMarshaller, 2479
 - createObject, 2479
 - getDataStructureType, 2479
 - looseMarshal, 2479
 - looseUnmarshal, 2480
 - ProducerInfoMarshaller, 2479
 - tightMarshal1, 2480
 - tightMarshal2, 2481
- activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller, 2552
 - ~RemoveInfoMarshaller, 2553
 - createObject, 2553
 - getDataStructureType, 2553
 - looseMarshal, 2553
 - looseUnmarshal, 2554

- RemoveInfoMarshaller, 2553
- tightMarshal1, 2554
- tightMarshal2, 2555
- tightUnmarshal, 2555
- activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller, 2576
 - ~RemoveSubscriptionInfoMarshaller, 2577
 - createObject, 2577
 - getDataStructureType, 2577
 - looseMarshal, 2577
 - looseUnmarshal, 2578
 - RemoveSubscriptionInfoMarshaller, 2577
 - tightMarshal1, 2578
 - tightMarshal2, 2579
 - tightUnmarshal, 2579
- activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller, 2591
 - ~ReplayCommandMarshaller, 2592
 - createObject, 2592
 - getDataStructureType, 2592
 - looseMarshal, 2593
 - looseUnmarshal, 2593
 - ReplayCommandMarshaller, 2592
 - tightMarshal1, 2593
 - tightMarshal2, 2594
 - tightUnmarshal, 2594
- activemq::wireformat::openwire::marshal::v3::ResponseInfoMarshaller, 2624
 - ~ResponseMarshaller, 2625
 - createObject, 2625
 - getDataStructureType, 2625
 - looseMarshal, 2626
 - looseUnmarshal, 2626
 - ResponseMarshaller, 2625
 - tightMarshal1, 2627
 - tightMarshal2, 2627
 - tightUnmarshal, 2628
- activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller, 2697
 - ~SessionIdMarshaller, 2699
 - createObject, 2699
 - getDataStructureType, 2699
 - looseMarshal, 2699
 - looseUnmarshal, 2699
 - SessionIdMarshaller, 2699
 - tightMarshal1, 2700
 - tightMarshal2, 2700
 - tightUnmarshal, 2701
- activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller, 2721
 - ~SessionInfoMarshaller, 2722
 - createObject, 2722
 - getDataStructureType, 2722
 - looseMarshal, 2722
 - looseUnmarshal, 2723
 - SessionInfoMarshaller, 2722
 - tightMarshal1, 2723
 - tightMarshal2, 2724
- activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller, 2767
 - ~ShutdownInfoMarshaller, 2769
 - createObject, 2769
 - getDataStructureType, 2769
 - looseMarshal, 2769
 - looseUnmarshal, 2769
 - ShutdownInfoMarshaller, 2769
 - tightMarshal1, 2770
 - tightMarshal2, 2770
- activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller, 2915
 - ~SubscriptionInfoMarshaller, 2916
 - createObject, 2916
 - getDataStructureType, 2916
 - looseMarshal, 2917
 - looseUnmarshal, 2917
 - SubscriptionInfoMarshaller, 2916
 - tightMarshal1, 2917
 - tightMarshal2, 2918
- activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller, 3033
 - ~TransactionIdMarshaller, 3034
 - looseMarshal, 3034
 - looseUnmarshal, 3035
 - tightMarshal1, 3035
 - tightMarshal2, 3036
 - tightUnmarshal, 3036
 - TransactionIdMarshaller, 3034
- activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller, 3046
 - ~TransactionInfoMarshaller, 3046
 - createObject, 3046
 - getDataStructureType, 3046
 - looseMarshal, 3046
 - looseUnmarshal, 3046
 - tightMarshal1, 3047
 - tightMarshal2, 3047
 - tightUnmarshal, 3048
 - TransactionInfoMarshaller, 3046
- activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller, 3180
 - ~WireFormatInfoMarshaller, 3180
 - createObject, 3180
 - getDataStructureType, 3180
 - looseMarshal, 3180
 - looseUnmarshal, 3180

- tightMarshal1, 3181
- tightMarshal2, 3181
- tightUnmarshal, 3182
- WireFormatInfoMarshaller, 3180
- activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller, 3201
 - ~XATransactionIdMarshaller, 3202
 - createObject, 3202
 - getDataStructureType, 3202
 - looseMarshal, 3202
 - looseUnmarshal, 3203
 - tightMarshal1, 3203
 - tightMarshal2, 3204
 - tightUnmarshal, 3204
 - XATransactionIdMarshaller, 3202
- activemq::wireformat::openwire::marshal::v4, 90
- activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMessageMarshaller, 154
 - ~ActiveMQBlobMessageMarshaller, 156
 - ActiveMQBlobMessageMarshaller, 156
 - createObject, 156
 - getDataStructureType, 156
 - looseMarshal, 156
 - looseUnmarshal, 156
 - tightMarshal1, 157
 - tightMarshal2, 157
 - tightUnmarshal, 158
- activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMessageMarshaller, 190
 - ~ActiveMQBytesMessageMarshaller, 191
 - ActiveMQBytesMessageMarshaller, 191
 - createObject, 191
 - getDataStructureType, 191
 - looseMarshal, 191
 - looseUnmarshal, 192
 - tightMarshal1, 192
 - tightMarshal2, 193
 - tightUnmarshal, 193
- activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller, 258
 - ~ActiveMQDestinationMarshaller, 259
 - ActiveMQDestinationMarshaller, 259
 - looseMarshal, 259
 - looseUnmarshal, 259
 - tightMarshal1, 260
 - tightMarshal2, 260
 - tightUnmarshal, 261
- activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller, 292
 - ~ActiveMQMapMessageMarshaller, 294
 - ActiveMQMapMessageMarshaller, 294
 - createObject, 294
 - getDataStructureType, 294
- looseMarshal, 294
- looseUnmarshal, 294
- tightMarshal1, 295
- tightMarshal2, 295
- tightUnmarshal, 296
- activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller, 315
 - ~ActiveMQMessageMarshaller, 316
 - ActiveMQMessageMarshaller, 316
 - createObject, 316
 - getDataStructureType, 316
 - looseMarshal, 316
 - looseUnmarshal, 317
 - tightMarshal1, 317
 - tightMarshal2, 318
 - tightUnmarshal, 318
- activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller, 356
 - ~ActiveMQObjectMessageMarshaller, 356
 - ActiveMQObjectMessageMarshaller, 356
 - createObject, 356
 - getDataStructureType, 356
 - looseMarshal, 356
 - looseUnmarshal, 357
 - tightMarshal1, 357
 - tightMarshal2, 358
 - tightUnmarshal, 358
- activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMessageMarshaller, 390
 - ~ActiveMQQueueMarshaller, 392
 - ActiveMQQueueMarshaller, 392
 - createObject, 392
 - getDataStructureType, 392
 - looseMarshal, 392
 - looseUnmarshal, 392
 - tightMarshal1, 393
 - tightMarshal2, 393
 - tightUnmarshal, 394
- activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller, 443
 - ~ActiveMQStreamMessageMarshaller, 445
 - ActiveMQStreamMessageMarshaller, 445
 - createObject, 445
 - getDataStructureType, 445
 - looseMarshal, 445
 - looseUnmarshal, 445
 - tightMarshal1, 446
 - tightMarshal2, 446
 - tightUnmarshal, 447
- activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller, 466
 - ~ActiveMQTempDestinationMarshaller, 467
 - ActiveMQTempDestinationMarshaller, 467

- looseMarshal, 467
- looseUnmarshal, 467
- tightMarshal1, 468
- tightMarshal2, 468
- tightUnmarshal, 469
- activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller, 489
 - ~ActiveMQTempQueueMarshaller, 490
 - ActiveMQTempQueueMarshaller, 490
 - createObject, 490
 - getDataStructureType, 490
 - looseMarshal, 490
 - looseUnmarshal, 491
 - tightMarshal1, 491
 - tightMarshal2, 492
 - tightUnmarshal, 492
- activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller, 513
 - ~ActiveMQTempTopicMarshaller, 514
 - ActiveMQTempTopicMarshaller, 514
 - createObject, 514
 - getDataStructureType, 514
 - looseMarshal, 514
 - looseUnmarshal, 515
 - tightMarshal1, 515
 - tightMarshal2, 516
 - tightUnmarshal, 516
- activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller, 537
 - ~ActiveMQTextMessageMarshaller, 539
 - ActiveMQTextMessageMarshaller, 539
 - createObject, 539
 - getDataStructureType, 539
 - looseMarshal, 539
 - looseUnmarshal, 539
 - tightMarshal1, 540
 - tightMarshal2, 540
 - tightUnmarshal, 541
- activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller, 561
 - ~ActiveMQTopicMarshaller, 562
 - ActiveMQTopicMarshaller, 562
 - createObject, 562
 - getDataStructureType, 562
 - looseMarshal, 563
 - looseUnmarshal, 563
 - tightMarshal1, 563
 - tightMarshal2, 564
 - tightUnmarshal, 564
- activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller, 616
 - ~BaseCommandMarshaller, 618
 - BaseCommandMarshaller, 618
 - looseMarshal, 618
 - looseUnmarshal, 619
 - tightMarshal1, 620
 - tightMarshal2, 621
 - tightUnmarshal, 622
- activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller, 703
 - ~BrokerIdMarshaller, 703
 - BrokerIdMarshaller, 703
 - createObject, 703
 - getDataStructureType, 703
 - looseMarshal, 703
 - looseUnmarshal, 703
 - tightMarshal1, 704
 - tightMarshal2, 704
 - tightUnmarshal, 705
- activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller, 731
 - ~BrokerInfoMarshaller, 731
 - BrokerInfoMarshaller, 731
 - createObject, 731
 - getDataStructureType, 731
 - looseMarshal, 731
 - looseUnmarshal, 731
 - tightMarshal1, 732
 - tightMarshal2, 732
 - tightUnmarshal, 733
- activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller, 1069
 - ~ConnectionControlMarshaller, 1069
 - ConnectionControlMarshaller, 1069
 - createObject, 1069
 - getDataStructureType, 1069
 - looseMarshal, 1069
 - looseUnmarshal, 1069
 - tightMarshal1, 1070
 - tightMarshal2, 1070
 - tightUnmarshal, 1071
- activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller, 1092
 - ~ConnectionErrorMarshaller, 1092
 - ConnectionErrorMarshaller, 1092
 - createObject, 1092
 - getDataStructureType, 1092
 - looseMarshal, 1093
 - looseUnmarshal, 1093
 - tightMarshal1, 1093
 - tightMarshal2, 1094
 - tightUnmarshal, 1094
- activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller, 1118
 - ~ConnectionIdMarshaller, 1118
 - ConnectionIdMarshaller, 1118
 - createObject, 1118
 - getDataStructureType, 1118

- looseMarshal, 1118
- looseUnmarshal, 1118
- tightMarshal1, 1119
- tightMarshal2, 1119
- tightUnmarshal, 1120
- activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller, 1138
 - ~ConnectionInfoMarshaller, 1140
 - ConnectionInfoMarshaller, 1140
 - createObject, 1140
 - getDataSetType, 1140
 - looseMarshal, 1140
 - looseUnmarshal, 1140
 - tightMarshal1, 1141
 - tightMarshal2, 1141
 - tightUnmarshal, 1142
- activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller, 1174
 - ~ConsumerControlMarshaller, 1176
 - ConsumerControlMarshaller, 1176
 - createObject, 1176
 - getDataSetType, 1176
 - looseMarshal, 1176
 - looseUnmarshal, 1176
 - tightMarshal1, 1177
 - tightMarshal2, 1177
 - tightUnmarshal, 1178
- activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller, 1198
 - ~ConsumerIdMarshaller, 1200
 - ConsumerIdMarshaller, 1200
 - createObject, 1200
 - getDataSetType, 1200
 - looseMarshal, 1200
 - looseUnmarshal, 1200
 - tightMarshal1, 1201
 - tightMarshal2, 1201
 - tightUnmarshal, 1202
- activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller, 1230
 - ~ConsumerInfoMarshaller, 1232
 - ConsumerInfoMarshaller, 1232
 - createObject, 1232
 - getDataSetType, 1232
 - looseMarshal, 1232
 - looseUnmarshal, 1232
 - tightMarshal1, 1233
 - tightMarshal2, 1233
 - tightUnmarshal, 1234
- activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller, 1250
 - ~ControlCommandMarshaller, 1251
 - ControlCommandMarshaller, 1251
 - createObject, 1251
 - getDataSetType, 1251
 - looseMarshal, 1252
 - looseUnmarshal, 1252
 - tightMarshal1, 1252
 - tightMarshal2, 1253
 - tightUnmarshal, 1253
- activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller, 1275
 - ~DataArrayResponseMarshaller, 1276
 - createObject, 1276
 - DataArrayResponseMarshaller, 1276
 - getDataSetType, 1276
 - looseMarshal, 1276
 - looseUnmarshal, 1277
 - tightMarshal1, 1277
 - tightMarshal2, 1278
 - tightUnmarshal, 1278
- activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller, 1322
 - ~DataResponseMarshaller, 1323
 - createObject, 1323
 - DataResponseMarshaller, 1323
 - getDataSetType, 1323
 - looseMarshal, 1323
 - looseUnmarshal, 1324
 - tightMarshal1, 1324
 - tightMarshal2, 1325
 - tightUnmarshal, 1325
- activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller, 1403
 - ~DestinationInfoMarshaller, 1404
 - createObject, 1404
 - DestinationInfoMarshaller, 1404
 - getDataSetType, 1404
 - looseMarshal, 1404
 - looseUnmarshal, 1405
 - tightMarshal1, 1405
 - tightMarshal2, 1406
 - tightUnmarshal, 1406
- activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller, 1427
 - ~DiscoveryEventMarshaller, 1429
 - createObject, 1429
 - DiscoveryEventMarshaller, 1429
 - getDataSetType, 1429
 - looseMarshal, 1429
 - looseUnmarshal, 1429
 - tightMarshal1, 1430
 - tightMarshal2, 1430
 - tightUnmarshal, 1431
- activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller, 1499
 - ~ExceptionResponseMarshaller, 1500
 - createObject, 1500

- ExceptionResponseMarshaller, 1500
- getDataStructureType, 1500
- looseMarshal, 1500
- looseUnmarshal, 1501
- tightMarshal1, 1501
- tightMarshal2, 1502
- tightUnmarshal, 1502
- activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller, 1587
 - ~FlushCommandMarshaller, 1589
 - createObject, 1589
 - FlushCommandMarshaller, 1589
 - getDataStructureType, 1589
 - looseMarshal, 1589
 - looseUnmarshal, 1589
 - tightMarshal1, 1590
 - tightMarshal2, 1590
 - tightUnmarshal, 1591
- activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller, 1681
 - ~IntegerResponseMarshaller, 1683
 - createObject, 1683
 - getDataStructureType, 1683
 - IntegerResponseMarshaller, 1683
 - looseMarshal, 1683
 - looseUnmarshal, 1684
 - tightMarshal1, 1684
 - tightMarshal2, 1684
 - tightUnmarshal, 1685
- activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller, 1725
 - ~JournalQueueAckMarshaller, 1727
 - createObject, 1727
 - getDataStructureType, 1727
 - JournalQueueAckMarshaller, 1727
 - looseMarshal, 1727
 - looseUnmarshal, 1727
 - tightMarshal1, 1728
 - tightMarshal2, 1728
 - tightUnmarshal, 1729
- activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller, 1758
 - ~JournalTopicAckMarshaller, 1760
 - createObject, 1760
 - getDataStructureType, 1760
 - JournalTopicAckMarshaller, 1760
 - looseMarshal, 1760
 - looseUnmarshal, 1760
 - tightMarshal1, 1761
 - tightMarshal2, 1761
 - tightUnmarshal, 1762
- activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller, 1773
 - ~JournalTraceMarshaller, 1775
 - createObject, 1775
 - getDataStructureType, 1775
 - JournalTraceMarshaller, 1775
 - looseMarshal, 1775
 - looseUnmarshal, 1775
 - tightMarshal1, 1776
 - tightMarshal2, 1776
 - tightUnmarshal, 1777
- activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller, 1801
 - ~JournalTransactionMarshaller, 1802
 - createObject, 1802
 - getDataStructureType, 1802
 - JournalTransactionMarshaller, 1802
 - looseMarshal, 1802
 - looseUnmarshal, 1803
 - tightMarshal1, 1803
 - tightMarshal2, 1803
 - tightUnmarshal, 1804
- activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller, 1827
 - ~KeepAliveInfoMarshaller, 1829
 - createObject, 1829
 - getDataStructureType, 1829
 - KeepAliveInfoMarshaller, 1829
 - looseMarshal, 1829
 - looseUnmarshal, 1829
 - tightMarshal1, 1830
 - tightMarshal2, 1830
 - tightUnmarshal, 1831
- activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller, 1854
 - ~LastPartialCommandMarshaller, 1856
 - createObject, 1856
 - getDataStructureType, 1856
 - LastPartialCommandMarshaller, 1856
 - looseMarshal, 1856
 - looseUnmarshal, 1857
 - tightMarshal1, 1857
 - tightMarshal2, 1857
 - tightUnmarshal, 1858
- activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller, 1886
 - ~LocalTransactionIdMarshaller, 1887
 - createObject, 1887
 - getDataStructureType, 1887
 - LocalTransactionIdMarshaller, 1887
 - looseMarshal, 1887
 - looseUnmarshal, 1888
 - tightMarshal1, 1888
 - tightMarshal2, 1889
 - tightUnmarshal, 1889
- activemq::wireformat::openwire::marshal::v4::MarshallerFactory, 1996

- ~MarshallerFactory, 1996
- configure, 1996
- activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller, 2072
- ~MessageAckMarshaller, 2073
- createObject, 2073
- getDataStructureType, 2073
- looseMarshal, 2073
- looseUnmarshal, 2074
- MessageAckMarshaller, 2073
- tightMarshal1, 2074
- tightMarshal2, 2075
- tightUnmarshal, 2075
- activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller, 2099
- ~MessageDispatchMarshaller, 2101
- createObject, 2101
- getDataStructureType, 2101
- looseMarshal, 2101
- looseUnmarshal, 2101
- MessageDispatchMarshaller, 2101
- tightMarshal1, 2102
- tightMarshal2, 2102
- tightUnmarshal, 2103
- activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller, 2124
- ~MessageDispatchNotificationMarshaller, 2125
- createObject, 2125
- getDataStructureType, 2125
- looseMarshal, 2126
- looseUnmarshal, 2126
- MessageDispatchNotificationMarshaller, 2125
- tightMarshal1, 2126
- tightMarshal2, 2127
- tightUnmarshal, 2127
- activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller, 2150
- ~MessageIdMarshaller, 2151
- createObject, 2151
- getDataStructureType, 2151
- looseMarshal, 2151
- looseUnmarshal, 2152
- MessageIdMarshaller, 2151
- tightMarshal1, 2152
- tightMarshal2, 2153
- tightUnmarshal, 2153
- activemq::wireformat::openwire::marshal::v4::MessageMarshaller, 2170
- ~MessageMarshaller, 2172
- looseMarshal, 2172
- looseUnmarshal, 2172
- MessageMarshaller, 2172
- tightMarshal1, 2173
- tightMarshal2, 2173
- tightUnmarshal, 2174
- activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller, 2223
- ~MessagePullMarshaller, 2224
- createObject, 2224
- getDataStructureType, 2224
- looseMarshal, 2224
- looseUnmarshal, 2225
- MessagePullMarshaller, 2224
- tightMarshal1, 2225
- tightMarshal2, 2226
- tightUnmarshal, 2226
- activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller, 2265
- ~NetworkBridgeFilterMarshaller, 2267
- createObject, 2267
- getDataStructureType, 2267
- looseMarshal, 2267
- looseUnmarshal, 2267
- NetworkBridgeFilterMarshaller, 2267
- tightMarshal1, 2268
- tightMarshal2, 2268
- tightUnmarshal, 2268
- activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller, 2330
- ~PartialCommandMarshaller, 2332
- createObject, 2332
- getDataStructureType, 2332
- looseMarshal, 2332
- looseUnmarshal, 2333
- PartialCommandMarshaller, 2332
- tightMarshal1, 2333
- tightMarshal2, 2333
- tightUnmarshal, 2334
- activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller, 2433
- ~ProducerAckMarshaller, 2433
- createObject, 2433
- getDataStructureType, 2433
- looseMarshal, 2433
- looseUnmarshal, 2433
- ProducerAckMarshaller, 2433
- tightMarshal1, 2434
- tightMarshal2, 2434
- tightUnmarshal, 2435
- activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller, 2467
- ~ProducerIdMarshaller, 2467
- createObject, 2467
- getDataStructureType, 2467
- looseMarshal, 2467
- looseUnmarshal, 2467

- ProducerIdMarshaller, 2467
 - tightMarshal1, 2468
 - tightMarshal2, 2468
 - tightUnmarshal, 2469
- activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller, 2486
 - ~ProducerInfoMarshaller, 2487
 - createObject, 2487
 - getDataStructureType, 2487
 - looseMarshal, 2487
 - looseUnmarshal, 2488
 - ProducerInfoMarshaller, 2487
 - tightMarshal1, 2488
 - tightMarshal2, 2489
 - tightUnmarshal, 2489
- activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller, 2556
 - ~RemoveInfoMarshaller, 2557
 - createObject, 2557
 - getDataStructureType, 2557
 - looseMarshal, 2557
 - looseUnmarshal, 2558
 - RemoveInfoMarshaller, 2557
 - tightMarshal1, 2558
 - tightMarshal2, 2559
 - tightUnmarshal, 2559
- activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller, 2580
 - ~RemoveSubscriptionInfoMarshaller, 2581
 - createObject, 2581
 - getDataStructureType, 2581
 - looseMarshal, 2581
 - looseUnmarshal, 2582
 - RemoveSubscriptionInfoMarshaller, 2581
 - tightMarshal1, 2582
 - tightMarshal2, 2583
 - tightUnmarshal, 2583
- activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller, 2599
 - ~ReplayCommandMarshaller, 2600
 - createObject, 2600
 - getDataStructureType, 2600
 - looseMarshal, 2600
 - looseUnmarshal, 2601
 - ReplayCommandMarshaller, 2600
 - tightMarshal1, 2601
 - tightMarshal2, 2602
 - tightUnmarshal, 2602
- activemq::wireformat::openwire::marshal::v4::ResponseMarshaller, 2637
 - ~ResponseMarshaller, 2639
 - createObject, 2639
 - getDataStructureType, 2639
 - looseMarshal, 2639
 - looseUnmarshal, 2640
 - ResponseMarshaller, 2639
 - tightMarshal1, 2640
 - tightMarshal2, 2641
- activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller, 2693
 - ~SessionIdMarshaller, 2695
 - createObject, 2695
 - getDataStructureType, 2695
 - looseMarshal, 2695
 - looseUnmarshal, 2695
 - SessionIdMarshaller, 2695
 - tightMarshal1, 2696
 - tightMarshal2, 2696
- activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller, 2713
 - ~SessionInfoMarshaller, 2714
 - createObject, 2714
 - getDataStructureType, 2714
 - looseMarshal, 2714
 - looseUnmarshal, 2715
 - SessionInfoMarshaller, 2714
 - tightMarshal1, 2715
 - tightMarshal2, 2716
- activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller, 2763
 - ~ShutdownInfoMarshaller, 2765
 - createObject, 2765
 - getDataStructureType, 2765
 - looseMarshal, 2765
 - looseUnmarshal, 2765
 - ShutdownInfoMarshaller, 2765
 - tightMarshal1, 2766
 - tightMarshal2, 2766
- activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller, 2923
 - ~SubscriptionInfoMarshaller, 2924
 - createObject, 2924
 - getDataStructureType, 2924
 - looseMarshal, 2924
 - looseUnmarshal, 2925
 - SubscriptionInfoMarshaller, 2924
 - tightMarshal1, 2925
 - tightMarshal2, 2925
- activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller, 3026
 - ~TransactionIdMarshaller, 3027
 - looseMarshal, 3027
 - looseUnmarshal, 3027

- tightMarshal1, 3028
- tightMarshal2, 3028
- tightUnmarshal, 3029
- TransactionIdMarshaller, 3027
- activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller, 3052
 - ~TransactionInfoMarshaller, 3054
 - createObject, 3054
 - getDataStructureType, 3054
 - looseMarshal, 3054
 - looseUnmarshal, 3054
 - tightMarshal1, 3055
 - tightMarshal2, 3055
 - tightUnmarshal, 3056
 - TransactionInfoMarshaller, 3054
- activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller, 3174
 - ~WireFormatInfoMarshaller, 3176
 - createObject, 3176
 - getDataStructureType, 3176
 - looseMarshal, 3176
 - looseUnmarshal, 3176
 - tightMarshal1, 3177
 - tightMarshal2, 3177
 - tightUnmarshal, 3178
 - WireFormatInfoMarshaller, 3176
- activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller, 3205
 - ~XATransactionIdMarshaller, 3206
 - createObject, 3206
 - getDataStructureType, 3206
 - looseMarshal, 3206
 - looseUnmarshal, 3207
 - tightMarshal1, 3207
 - tightMarshal2, 3208
 - tightUnmarshal, 3208
 - XATransactionIdMarshaller, 3206
- activemq::wireformat::openwire::marshal::v5, 94
- activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller, 158
 - ~ActiveMQBlobMessageMarshaller, 160
 - ActiveMQBlobMessageMarshaller, 160
 - createObject, 160
 - getDataStructureType, 160
 - looseMarshal, 160
 - looseUnmarshal, 160
 - tightMarshal1, 161
 - tightMarshal2, 161
 - tightUnmarshal, 162
- activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMessageMarshaller, 194
 - ~ActiveMQBytesMessageMarshaller, 195
 - ActiveMQBytesMessageMarshaller, 195
- createObject, 195
- getDataStructureType, 195
- looseMarshal, 195
- looseUnmarshal, 196
- tightMarshal1, 196
- tightMarshal2, 197
- tightUnmarshal, 197
- activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller, 262
 - ~ActiveMQDestinationMarshaller, 263
 - ActiveMQDestinationMarshaller, 263
 - looseMarshal, 263
 - looseUnmarshal, 263
 - tightMarshal1, 264
 - tightMarshal2, 264
 - tightUnmarshal, 265
- activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller, 296
 - ~ActiveMQMapMessageMarshaller, 298
 - ActiveMQMapMessageMarshaller, 298
 - createObject, 298
 - getDataStructureType, 298
 - looseMarshal, 298
 - looseUnmarshal, 298
 - tightMarshal1, 299
 - tightMarshal2, 299
 - tightUnmarshal, 300
- activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller, 319
 - ~ActiveMQMessageMarshaller, 320
 - ActiveMQMessageMarshaller, 320
 - createObject, 320
 - getDataStructureType, 320
 - looseMarshal, 320
 - looseUnmarshal, 321
 - tightMarshal1, 321
 - tightMarshal2, 322
 - tightUnmarshal, 322
- activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller, 339
 - ~ActiveMQObjectMessageMarshaller, 360
 - ActiveMQObjectMessageMarshaller, 360
 - createObject, 360
 - getDataStructureType, 360
 - looseMarshal, 360
 - looseUnmarshal, 361
 - tightMarshal1, 361
 - tightMarshal2, 362
 - tightUnmarshal, 362
- activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMessageMarshaller, 396
 - ~ActiveMQQueueMarshaller, 396
 - ActiveMQQueueMarshaller, 396
 - createObject, 396

- getDataStructureType, 396
 - looseMarshal, 396
 - looseUnmarshal, 396
 - tightMarshal1, 397
 - tightMarshal2, 397
 - tightUnmarshal, 398
- activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller, 447
 - ~ActiveMQStreamMessageMarshaller, 449
 - ActiveMQStreamMessageMarshaller, 449
 - createObject, 449
 - getDataStructureType, 449
 - looseMarshal, 449
 - looseUnmarshal, 449
 - tightMarshal1, 450
 - tightMarshal2, 450
 - tightUnmarshal, 451
- activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller, 470
 - ~ActiveMQTempDestinationMarshaller, 471
 - ActiveMQTempDestinationMarshaller, 471
 - looseMarshal, 471
 - looseUnmarshal, 471
 - tightMarshal1, 472
 - tightMarshal2, 472
 - tightUnmarshal, 473
- activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller, 493
 - ~ActiveMQTempQueueMarshaller, 494
 - ActiveMQTempQueueMarshaller, 494
 - createObject, 494
 - getDataStructureType, 494
 - looseMarshal, 494
 - looseUnmarshal, 495
 - tightMarshal1, 495
 - tightMarshal2, 496
 - tightUnmarshal, 496
- activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller, 517
 - ~ActiveMQTempTopicMarshaller, 518
 - ActiveMQTempTopicMarshaller, 518
 - createObject, 518
 - getDataStructureType, 518
 - looseMarshal, 518
 - looseUnmarshal, 519
 - tightMarshal1, 519
 - tightMarshal2, 520
 - tightUnmarshal, 520
- activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller, 541
 - ~ActiveMQTextMessageMarshaller, 543
 - ActiveMQTextMessageMarshaller, 543
 - createObject, 543
- getDataStructureType, 543
 - looseMarshal, 543
 - looseUnmarshal, 543
 - tightMarshal1, 544
 - tightMarshal2, 544
 - tightUnmarshal, 545
- activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller, 565
 - ~ActiveMQTopicMarshaller, 566
 - ActiveMQTopicMarshaller, 566
 - createObject, 566
 - getDataStructureType, 566
 - looseMarshal, 566
 - looseUnmarshal, 567
 - tightMarshal1, 567
 - tightMarshal2, 568
 - tightUnmarshal, 568
- activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller, 623
 - ~BaseCommandMarshaller, 624
 - BaseCommandMarshaller, 624
 - looseMarshal, 624
 - looseUnmarshal, 625
 - tightMarshal1, 626
 - tightMarshal2, 627
 - tightUnmarshal, 629
- activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller, 700
 - ~BrokerIdMarshaller, 707
 - BrokerIdMarshaller, 707
 - createObject, 707
 - getDataStructureType, 707
 - looseMarshal, 707
 - looseUnmarshal, 707
 - tightMarshal1, 708
 - tightMarshal2, 708
 - tightUnmarshal, 709
- activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller, 731
 - ~BrokerInfoMarshaller, 735
 - BrokerInfoMarshaller, 735
 - createObject, 735
 - getDataStructureType, 735
 - looseMarshal, 735
 - looseUnmarshal, 735
 - tightMarshal1, 736
 - tightMarshal2, 736
 - tightUnmarshal, 737
- activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller, 1071
 - ~ConnectionControlMarshaller, 1073
 - ConnectionControlMarshaller, 1073
 - createObject, 1073
 - getDataStructureType, 1073

- looseMarshal, 1073
- looseUnmarshal, 1073
- tightMarshal1, 1074
- tightMarshal2, 1074
- tightUnmarshal, 1075
- activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller, 1095
 - ~ConnectionErrorMarshaller, 1096
 - ConnectionErrorMarshaller, 1096
 - createObject, 1096
 - getDataStructureType, 1096
 - looseMarshal, 1096
 - looseUnmarshal, 1097
 - tightMarshal1, 1097
 - tightMarshal2, 1098
 - tightUnmarshal, 1098
- activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller, 1120
 - ~ConnectionIdMarshaller, 1122
 - ConnectionIdMarshaller, 1122
 - createObject, 1122
 - getDataStructureType, 1122
 - looseMarshal, 1122
 - looseUnmarshal, 1122
 - tightMarshal1, 1123
 - tightMarshal2, 1123
 - tightUnmarshal, 1124
- activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller, 1146
 - ~ConnectionInfoMarshaller, 1148
 - ConnectionInfoMarshaller, 1148
 - createObject, 1148
 - getDataStructureType, 1148
 - looseMarshal, 1148
 - looseUnmarshal, 1148
 - tightMarshal1, 1149
 - tightMarshal2, 1149
 - tightUnmarshal, 1150
- activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller, 1182
 - ~ConsumerControlMarshaller, 1184
 - ConsumerControlMarshaller, 1184
 - createObject, 1184
 - getDataStructureType, 1184
 - looseMarshal, 1184
 - looseUnmarshal, 1184
 - tightMarshal1, 1185
 - tightMarshal2, 1185
 - tightUnmarshal, 1186
- activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller, 1206
 - ~ConsumerIdMarshaller, 1208
 - ConsumerIdMarshaller, 1208
 - createObject, 1208
 - getDataStructureType, 1208
 - looseMarshal, 1208
 - looseUnmarshal, 1208
 - tightMarshal1, 1209
 - tightMarshal2, 1209
 - tightUnmarshal, 1210
- activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller, 1234
 - ~ConsumerInfoMarshaller, 1236
 - ConsumerInfoMarshaller, 1236
 - createObject, 1236
 - getDataStructureType, 1236
 - looseMarshal, 1236
 - looseUnmarshal, 1236
 - tightMarshal1, 1237
 - tightMarshal2, 1237
 - tightUnmarshal, 1238
- activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller, 1258
 - ~ControlCommandMarshaller, 1259
 - ControlCommandMarshaller, 1259
 - createObject, 1259
 - getDataStructureType, 1259
 - looseMarshal, 1259
 - looseUnmarshal, 1260
 - tightMarshal1, 1260
 - tightMarshal2, 1261
 - tightUnmarshal, 1261
- activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller, 1283
 - ~DataArrayResponseMarshaller, 1284
 - createObject, 1284
 - DataArrayResponseMarshaller, 1284
 - getDataStructureType, 1284
 - looseMarshal, 1284
 - looseUnmarshal, 1285
 - tightMarshal1, 1285
 - tightMarshal2, 1286
 - tightUnmarshal, 1286
- activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller, 1318
 - ~DataResponseMarshaller, 1319
 - createObject, 1319
 - DataResponseMarshaller, 1319
 - getDataStructureType, 1319
 - looseMarshal, 1319
 - looseUnmarshal, 1320
 - tightMarshal1, 1320
 - tightMarshal2, 1321
 - tightUnmarshal, 1321
- activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller, 1411
 - ~DestinationInfoMarshaller, 1412
 - createObject, 1412

- DestinationInfoMarshaller, 1412
- getDataStructureType, 1412
- looseMarshal, 1412
- looseUnmarshal, 1413
- tightMarshal1, 1413
- tightMarshal2, 1414
- tightUnmarshal, 1414
- activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller, 1431
 - ~DiscoveryEventMarshaller, 1433
 - createObject, 1433
 - DiscoveryEventMarshaller, 1433
 - getDataStructureType, 1433
 - looseMarshal, 1433
 - looseUnmarshal, 1433
 - tightMarshal1, 1434
 - tightMarshal2, 1434
 - tightUnmarshal, 1435
- activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller, 1503
 - ~ExceptionResponseMarshaller, 1504
 - createObject, 1504
 - ExceptionResponseMarshaller, 1504
 - getDataStructureType, 1504
 - looseMarshal, 1504
 - looseUnmarshal, 1505
 - tightMarshal1, 1505
 - tightMarshal2, 1506
 - tightUnmarshal, 1506
- activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller, 1591
 - ~FlushCommandMarshaller, 1593
 - createObject, 1593
 - FlushCommandMarshaller, 1593
 - getDataStructureType, 1593
 - looseMarshal, 1593
 - looseUnmarshal, 1593
 - tightMarshal1, 1594
 - tightMarshal2, 1594
 - tightUnmarshal, 1595
- activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller, 1685
 - ~IntegerResponseMarshaller, 1687
 - createObject, 1687
 - getDataStructureType, 1687
 - IntegerResponseMarshaller, 1687
 - looseMarshal, 1687
 - looseUnmarshal, 1688
 - tightMarshal1, 1688
 - tightMarshal2, 1688
 - tightUnmarshal, 1689
- activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller, 1737
 - ~JournalQueueAckMarshaller, 1739
- createObject, 1739
- getDataStructureType, 1739
- JournalQueueAckMarshaller, 1739
- looseMarshal, 1739
- looseUnmarshal, 1739
- tightMarshal1, 1740
- tightMarshal2, 1740
- activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller, 1762
 - ~JournalTopicAckMarshaller, 1764
 - createObject, 1764
 - getDataStructureType, 1764
 - JournalTopicAckMarshaller, 1764
 - looseMarshal, 1764
 - looseUnmarshal, 1764
 - tightMarshal1, 1765
 - tightMarshal2, 1765
- activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller, 1785
 - ~JournalTraceMarshaller, 1787
 - createObject, 1787
 - getDataStructureType, 1787
 - JournalTraceMarshaller, 1787
 - looseMarshal, 1787
 - looseUnmarshal, 1787
 - tightMarshal1, 1788
 - tightMarshal2, 1788
- activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller, 1808
 - ~JournalTransactionMarshaller, 1810
 - createObject, 1810
 - getDataStructureType, 1810
 - JournalTransactionMarshaller, 1810
 - looseMarshal, 1810
 - looseUnmarshal, 1810
 - tightMarshal1, 1811
 - tightMarshal2, 1811
- activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller, 1819
 - ~KeepAliveInfoMarshaller, 1821
 - createObject, 1821
 - getDataStructureType, 1821
 - KeepAliveInfoMarshaller, 1821
 - looseMarshal, 1821
 - looseUnmarshal, 1821
 - tightMarshal1, 1822
 - tightMarshal2, 1822
- activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller, 1858
 - ~LastPartialCommandMarshaller, 1860

- ~LastPartialCommandMarshaller, 1860
- createObject, 1860
- getDataStructureType, 1860
- LastPartialCommandMarshaller, 1860
- looseMarshal, 1860
- looseUnmarshal, 1861
- tightMarshal1, 1861
- tightMarshal2, 1861
- tightUnmarshal, 1862
- activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller, 1894
 - ~LocalTransactionIdMarshaller, 1895
 - createObject, 1895
 - getDataStructureType, 1895
 - LocalTransactionIdMarshaller, 1895
 - looseMarshal, 1895
 - looseUnmarshal, 1896
 - tightMarshal1, 1896
 - tightMarshal2, 1897
 - tightUnmarshal, 1897
- activemq::wireformat::openwire::marshal::v5::MarshallerFactory, 1998
 - ~MarshallerFactory, 1999
 - configure, 1999
- activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller, 2064
 - ~MessageAckMarshaller, 2065
 - createObject, 2065
 - getDataStructureType, 2065
 - looseMarshal, 2065
 - looseUnmarshal, 2066
 - MessageAckMarshaller, 2065
 - tightMarshal1, 2066
 - tightMarshal2, 2067
 - tightUnmarshal, 2067
- activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller, 2111
 - ~MessageDispatchMarshaller, 2113
 - createObject, 2113
 - getDataStructureType, 2113
 - looseMarshal, 2113
 - looseUnmarshal, 2113
 - MessageDispatchMarshaller, 2113
 - tightMarshal1, 2114
 - tightMarshal2, 2114
 - tightUnmarshal, 2115
- activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller, 2136
 - ~MessageDispatchNotificationMarshaller, 2137
 - createObject, 2137
 - getDataStructureType, 2137
 - looseMarshal, 2137
 - looseUnmarshal, 2138
- MessageDispatchNotificationMarshaller, 2137
- tightMarshal1, 2138
- tightMarshal2, 2139
- tightUnmarshal, 2139
- activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller, 2157
 - ~MessageIdMarshaller, 2159
 - createObject, 2159
 - getDataStructureType, 2159
 - looseMarshal, 2159
 - looseUnmarshal, 2159
 - MessageIdMarshaller, 2159
 - tightMarshal1, 2160
 - tightMarshal2, 2160
 - tightUnmarshal, 2161
- activemq::wireformat::openwire::marshal::v5::MessageMarshaller, 2179
 - ~MessageMarshaller, 2180
 - looseMarshal, 2180
 - looseUnmarshal, 2180
 - MessageMarshaller, 2180
 - tightMarshal1, 2181
 - tightMarshal2, 2181
 - tightUnmarshal, 2182
- activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller, 2219
 - ~MessagePullMarshaller, 2220
 - createObject, 2220
 - getDataStructureType, 2220
 - looseMarshal, 2220
 - looseUnmarshal, 2221
 - MessagePullMarshaller, 2220
 - tightMarshal1, 2221
 - tightMarshal2, 2222
- activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller, 2222
 - ~MessagePullMarshaller, 2222
- activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller, 2257
 - ~NetworkBridgeFilterMarshaller, 2259
 - createObject, 2259
 - getDataStructureType, 2259
 - looseMarshal, 2259
 - looseUnmarshal, 2259
 - NetworkBridgeFilterMarshaller, 2259
 - tightMarshal1, 2260
 - tightMarshal2, 2260
- activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller, 2339
 - ~PartialCommandMarshaller, 2340
 - createObject, 2340
 - getDataStructureType, 2340
 - looseMarshal, 2340
 - looseUnmarshal, 2341

- PartialCommandMarshaller, 2340
- tightMarshal1, 2341
- tightMarshal2, 2342
- tightUnmarshal, 2342
- activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller, 2435
 - ~ProducerAckMarshaller, 2437
 - createObject, 2437
 - getDataStructureType, 2437
 - looseMarshal, 2437
 - looseUnmarshal, 2437
 - ProducerAckMarshaller, 2437
 - tightMarshal1, 2438
 - tightMarshal2, 2438
 - tightUnmarshal, 2439
- activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller, 2457
 - ~ProducerIdMarshaller, 2459
 - createObject, 2459
 - getDataStructureType, 2459
 - looseMarshal, 2459
 - looseUnmarshal, 2459
 - ProducerIdMarshaller, 2459
 - tightMarshal1, 2460
 - tightMarshal2, 2460
 - tightUnmarshal, 2461
- activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller, 2482
 - ~ProducerInfoMarshaller, 2483
 - createObject, 2483
 - getDataStructureType, 2483
 - looseMarshal, 2483
 - looseUnmarshal, 2484
 - ProducerInfoMarshaller, 2483
 - tightMarshal1, 2484
 - tightMarshal2, 2485
 - tightUnmarshal, 2485
- activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller, 2544
 - ~RemoveInfoMarshaller, 2545
 - createObject, 2545
 - getDataStructureType, 2545
 - looseMarshal, 2545
 - looseUnmarshal, 2546
 - RemoveInfoMarshaller, 2545
 - tightMarshal1, 2546
 - tightMarshal2, 2547
 - tightUnmarshal, 2547
- activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller, 2564
 - ~RemoveSubscriptionInfoMarshaller, 2565
 - createObject, 2565
 - getDataStructureType, 2566
 - looseMarshal, 2566
 - looseUnmarshal, 2566
 - RemoveSubscriptionInfoMarshaller, 2565
 - tightMarshal1, 2566
 - tightMarshal2, 2567
 - tightUnmarshal, 2567
- activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller, 2595
 - ~ReplayCommandMarshaller, 2596
 - createObject, 2596
 - getDataStructureType, 2596
 - looseMarshal, 2596
 - looseUnmarshal, 2597
 - ReplayCommandMarshaller, 2596
 - tightMarshal1, 2597
 - tightMarshal2, 2598
 - tightUnmarshal, 2598
- activemq::wireformat::openwire::marshal::v5::ResponseMarshaller, 2628
 - ~ResponseMarshaller, 2630
 - createObject, 2630
 - getDataStructureType, 2630
 - looseMarshal, 2630
 - looseUnmarshal, 2631
 - ResponseMarshaller, 2630
 - tightMarshal1, 2631
 - tightMarshal2, 2632
 - tightUnmarshal, 2632
- activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller, 2681
 - ~SessionIdMarshaller, 2683
 - createObject, 2683
 - getDataStructureType, 2683
 - looseMarshal, 2683
 - looseUnmarshal, 2683
 - SessionIdMarshaller, 2683
 - tightMarshal1, 2684
 - tightMarshal2, 2684
 - tightUnmarshal, 2685
- activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller, 2717
 - ~SessionInfoMarshaller, 2718
 - createObject, 2718
 - getDataStructureType, 2718
 - looseMarshal, 2718
 - looseUnmarshal, 2719
 - SessionInfoMarshaller, 2718
 - tightMarshal1, 2719
 - tightMarshal2, 2720
 - tightUnmarshal, 2720
- activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller, 2775
 - ~ShutdownInfoMarshaller, 2777
 - createObject, 2777
 - getDataStructureType, 2777

- looseMarshal, 2777
- looseUnmarshal, 2777
- ShutdownInfoMarshaller, 2777
- tightMarshal1, 2778
- tightMarshal2, 2778
- tightUnmarshal, 2779
- activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller, 2919
 - ~SubscriptionInfoMarshaller, 2920
 - createObject, 2920
 - getDataStructureType, 2920
 - looseMarshal, 2920
 - looseUnmarshal, 2921
 - SubscriptionInfoMarshaller, 2920
 - tightMarshal1, 2921
 - tightMarshal2, 2922
 - tightUnmarshal, 2922
- activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller, 3029
 - ~TransactionIdMarshaller, 3031
 - looseMarshal, 3031
 - looseUnmarshal, 3031
 - tightMarshal1, 3032
 - tightMarshal2, 3032
 - tightUnmarshal, 3033
 - TransactionIdMarshaller, 3031
- activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller, 3041
 - ~TransactionInfoMarshaller, 3042
 - createObject, 3042
 - getDataStructureType, 3042
 - looseMarshal, 3042
 - looseUnmarshal, 3043
 - tightMarshal1, 3043
 - tightMarshal2, 3043
 - tightUnmarshal, 3044
 - TransactionInfoMarshaller, 3042
- activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller, 3166
 - ~WireFormatInfoMarshaller, 3168
 - createObject, 3168
 - getDataStructureType, 3168
 - looseMarshal, 3168
 - looseUnmarshal, 3168
 - tightMarshal1, 3169
 - tightMarshal2, 3169
 - tightUnmarshal, 3170
 - WireFormatInfoMarshaller, 3168
- activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller, 3213
 - ~XATransactionIdMarshaller, 3214
 - createObject, 3214
 - getDataStructureType, 3214
 - looseMarshal, 3214
 - looseUnmarshal, 3215
 - tightMarshal1, 3215
 - tightMarshal2, 3216
 - tightUnmarshal, 3216
 - XATransactionIdMarshaller, 3214
- activemq::wireformat::openwire::OpenWireFormat, 2299
 - ~OpenWireFormat, 2299
 - addMarshaller, 2300
 - createNegotiator, 2300
 - DEFAULT_VERSION, 2308
 - destroyMarshallers, 2300
 - doUnmarshal, 2300
 - getCacheSize, 2301
 - getMaxInactivityDuration, 2301
 - getMaxInactivityDurationInitialDelay, 2301
 - getPinCodeWireFormatInfo, 2301
 - getVersion, 2301
 - hasNegotiator, 2302
 - inReceive, 2302
 - isCacheEnabled, 2302
 - isSizePrefixDisabled, 2302
 - isStackTraceEnabled, 2302
 - isTcpNoDelayEnabled, 2303
 - isTightEncodingEnabled, 2303
 - looseMarshalNestedObject, 2303
 - looseUnmarshalNestedObject, 2303
 - marshal, 2304
 - NULL_TYPE, 2308
 - OpenWireFormat, 2299
 - renegotiateWireFormat, 2304
 - setCacheEnabled, 2305
 - setCacheSize, 2305
 - setMaxInactivityDuration, 2305
 - setMaxInactivityDurationInitialDelay, 2305
 - setPinCodeWireFormatInfo, 2305
 - setSizePrefixDisabled, 2305
 - setStackTraceEnabled, 2306
 - setTcpNoDelayEnabled, 2306
 - setTightEncodingEnabled, 2306
 - setVersion, 2306
 - tightMarshalNestedObject1, 2306
 - tightMarshalNestedObject2, 2307
 - tightUnmarshalNestedObject, 2307
 - unmarshal, 2308
- activemq::wireformat::openwire::OpenWireFormatFactory, 2308
 - ~OpenWireFormatFactory, 2309
 - createWireFormat, 2309
 - OpenWireFormatFactory, 2309
- activemq::wireformat::openwire::OpenWireFormatNegotiator, 2310

- ~OpenWireFormatNegotiator, 2311
- close, 2311
- onCommand, 2311
- oneway, 2311
- onTransportException, 2312
- OpenWireFormatNegotiator, 2310
- request, 2312
- start, 2313
- activemq::wireformat::openwire::OpenWireResponseBuilder, 2313
 - ~OpenWireResponseBuilder, 2314
 - buildIncomingCommands, 2314
 - buildResponse, 2314
 - OpenWireResponseBuilder, 2314
- activemq::wireformat::openwire::utils, 98
- activemq::wireformat::openwire::utils::BooleanStream, 680
 - ~BooleanStream, 682
 - BooleanStream, 682
 - clear, 682
 - marshal, 682
 - marshalledSize, 682
 - readBoolean, 682
 - unmarshal, 682
 - writeBoolean, 683
- activemq::wireformat::openwire::utils::HexTable, 1609
 - ~HexTable, 1610
 - HexTable, 1610
 - operator[], 1610
 - size, 1610
- activemq::wireformat::openwire::utils::MessagePropertyInterceptor, 2196
 - ~MessagePropertyInterceptor, 2198
 - getBooleanProperty, 2198
 - getByteProperty, 2198
 - getDoubleProperty, 2199
 - getFloatProperty, 2199
 - getIntProperty, 2199
 - getLongProperty, 2199
 - getShortProperty, 2200
 - getStringProperty, 2200
 - MessagePropertyInterceptor, 2198
 - setBooleanProperty, 2200
 - setByteProperty, 2200
 - setDoubleProperty, 2201
 - setFloatProperty, 2201
 - setIntProperty, 2201
 - setLongProperty, 2201
 - setShortProperty, 2202
 - setStringProperty, 2202
- activemq::wireformat::openwire::utils::OpenwireStringSupport, 2315
 - ~OpenwireStringSupport, 2315
 - readString, 2315
 - writeString, 2316
- activemq::wireformat::stomp, 99
- activemq::wireformat::stomp::StompCommandConstants, 2871
 - ABORT, 2873
 - ACK, 2873
 - ACK_AUTO, 2873
 - ACK_CLIENT, 2873
 - ACK_INDIVIDUAL, 2873
 - BEGIN, 2873
 - BYTES, 2873
 - COMMIT, 2873
 - CONNECT, 2873
 - CONNECTED, 2873
 - DISCONNECT, 2873
 - ERROR_CMD, 2873
 - HEADER_ACK, 2873
 - HEADER_CLIENT_ID, 2873
 - HEADER_CONSUMERPRIORITY, 2873
 - HEADER_CONTENTLENGTH, 2873
 - HEADER_CORRELATIONID, 2873
 - HEADER_DESTINATION, 2873
 - HEADER_DISPATCH_ASYNC, 2873
 - HEADER_EXCLUSIVE, 2873
 - HEADER_EXPIRES, 2873
 - HEADER_ID, 2873
 - HEADER_JMSPRIORITY, 2873
 - HEADER_LOGIN, 2873
 - HEADER_MAXPENDINGMSGLIMIT, 2873
 - HEADER_MESSAGE, 2873
 - HEADER_MESSAGEID, 2873
 - HEADER_NOLOCAL, 2873
 - HEADER_OLDSUBSCRIPTIONNAME, 2873
 - HEADER_PASSWORD, 2873
 - HEADER_PERSISTENT, 2873
 - HEADER_PREFETCHSIZE, 2873
 - HEADER_RECEIPT_REQUIRED, 2873
 - HEADER_RECEIPTID, 2873
 - HEADER_REDELIVERED, 2873
 - HEADER_REDELIVERYCOUNT, 2873
 - HEADER_REPLYTO, 2873
 - HEADER_REQUESTID, 2873
 - HEADER_RESPONSEID, 2873
 - HEADER_RETROACTIVE, 2873
 - HEADER_SELECTOR, 2873
 - HEADER_SESSIONID, 2873
 - HEADER_SUBSCRIPTION, 2873
 - HEADER_SUBSCRIPTIONNAME, 2873
 - HEADER_TIMESTAMP, 2873
 - HEADER_TRANSACTIONID, 2873

- HEADER_TRANSFORMATION, 2873
- HEADER_TRANSFORMATION_-
ERROR, 2873
- HEADER_TYPE, 2873
- MESSAGE, 2873
- QUEUE_PREFIX, 2873
- RECEIPT, 2873
- SEND, 2873
- SUBSCRIBE, 2873
- TEMPQUEUE_PREFIX, 2873
- TEMPTOPIC_PREFIX, 2873
- TEXT, 2873
- TOPIC_PREFIX, 2873
- UNSUBSCRIBE, 2873
- activemq::wireformat::stomp::StompFrame,
2874
 - ~StompFrame, 2875
 - clone, 2875
 - copy, 2875
 - fromStream, 2876
 - getBody, 2876
 - getBodyLength, 2876
 - getCommand, 2876
 - getProperties, 2876, 2877
 - getProperty, 2877
 - hasProperty, 2877
 - removeProperty, 2877
 - setBody, 2877
 - setCommand, 2878
 - setProperty, 2878
 - StompFrame, 2875
 - toStream, 2878
- activemq::wireformat::stomp::StompHelper,
2878
 - ~StompHelper, 2880
 - convertConsumerId, 2880
 - convertDestination, 2880
 - convertMessageId, 2881
 - convertProducerId, 2881
 - convertProperties, 2882
 - convertTransactionId, 2882
 - StompHelper, 2880
- activemq::wireformat::stomp::StompWireFormat,
2883
 - ~StompWireFormat, 2884
 - createNegotiator, 2884
 - getVersion, 2884
 - hasNegotiator, 2884
 - inReceive, 2885
 - marshal, 2885
 - setVersion, 2885
 - StompWireFormat, 2884
 - unmarshal, 2885
- activemq::wireformat::stomp::StompWireFormatFactory,
2886
 - ~StompWireFormatFactory, 2887
 - createWireFormat, 2887
 - StompWireFormatFactory, 2887
- activemq::wireformat::WireFormat, 3148
 - ~WireFormat, 3149
 - createNegotiator, 3149
 - getVersion, 3149
 - hasNegotiator, 3149
 - inReceive, 3150
 - marshal, 3150
 - setVersion, 3150
 - unmarshal, 3151
- activemq::wireformat::WireFormatFactory,
3151
 - ~WireFormatFactory, 3152
 - createWireFormat, 3152
- activemq::wireformat::WireFormatNegotiator,
3182
 - ~WireFormatNegotiator, 3183
 - WireFormatNegotiator, 3183
- activemq::wireformat::WireFormatRegistry,
3183
 - ~WireFormatRegistry, 3184
 - findFactory, 3184
 - getInstance, 3184
 - getWireFormatNames, 3185
 - registerFactory, 3185
 - unregisterFactory, 3185
- ActiveMQBlobMessage
 - activemq::commands::ActiveMQBlobMessage,
143
- ActiveMQBlobMessageMarshaller
 - activemq::wireformat::openwire::marshal::v1::ActiveMQBlobM
152
 - activemq::wireformat::openwire::marshal::v2::ActiveMQBlobM
164
 - activemq::wireformat::openwire::marshal::v3::ActiveMQBlobM
148
 - activemq::wireformat::openwire::marshal::v4::ActiveMQBlobM
156
 - activemq::wireformat::openwire::marshal::v5::ActiveMQBlobM
160
- ActiveMQBytesMessage
 - activemq::commands::ActiveMQBytesMessage,
170
- ActiveMQBytesMessageMarshaller
 - activemq::wireformat::openwire::marshal::v1::ActiveMQBytesM
187
 - activemq::wireformat::openwire::marshal::v2::ActiveMQBytesM
199
 - activemq::wireformat::openwire::marshal::v3::ActiveMQBytesM
183

- activemq::wireformat::openwire::marshal::v4::ActiveMQByteMessageMarshaller, 191
- activemq::wireformat::openwire::marshal::v5::ActiveMQByteMessageMarshaller, 195
- ActiveMQConnection
 - activemq::core::ActiveMQConnection, 204
- ActiveMQConnectionFactory
 - activemq::core::ActiveMQConnectionFactory, 213
- ActiveMQConnectionMetaData
 - activemq::core::ActiveMQConnectionMetaData, 217
- ActiveMQConnectionSupport
 - activemq::core::ActiveMQConnectionSupport, 222
- ActiveMQConsumer
 - activemq::core::ActiveMQConsumer, 232
- ActiveMQCPP
 - activemq::library::ActiveMQCPP, 239
- ActiveMQDestination
 - activemq::commands::ActiveMQDestination, 242
- ActiveMQDestinationMarshaller
 - activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller, 255
 - activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller, 267
 - activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller, 251
 - activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller, 259
 - activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller, 263
- ActiveMQException
 - activemq::exceptions::ActiveMQException, 270
- ActiveMQMapMessage
 - activemq::commands::ActiveMQMapMessage, 275
- ActiveMQMapMessageMarshaller
 - activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller, 290
 - activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller, 302
 - activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller, 286
 - activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller, 294
 - activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller, 298
- ActiveMQMessage
 - activemq::commands::ActiveMQMessage, 305
- ActiveMQMessageMarshaller
 - activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller, 312
 - activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller, 324
 - activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller, 308
 - activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller, 316
 - activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller, 320
- ActiveMQMessageTemplate
 - activemq::commands::ActiveMQMessageTemplate, 331
- ActiveMQObjectMessage
 - activemq::commands::ActiveMQObjectMessage, 345
- ActiveMQObjectMessageMarshaller
 - activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller, 352
 - activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller, 364
 - activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller, 348
 - activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller, 356
 - activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller, 360
- ActiveMQProducer
 - activemq::core::ActiveMQProducer, 369
- ActiveMQQueue
 - activemq::commands::ActiveMQQueue, 380
- ActiveMQQueueMarshaller
 - activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller, 388
 - activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller, 400
 - activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller, 384
 - activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller, 392
 - activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller, 396
- ActiveMQSession
 - activemq::core::ActiveMQSession, 406
- ActiveMQSessionExecutor
 - activemq::core::ActiveMQSession, 418
- ActiveMQSessionExecutor
 - activemq::core::ActiveMQSession, 419
- ActiveMQStreamMessage
 - activemq::commands::ActiveMQStreamMessage, 425
- ActiveMQStreamMessageMarshaller
 - activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller, 432
 - activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller, 444
 - activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller, 428
 - activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller, 436
 - activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller, 440

- activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller, 441
- activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller, 453
- activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller, 437
- activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMessageMarshaller, 445
- activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller, 449
- ActiveMQTempDestination
 - activemq::commands::ActiveMQTempDestination, 457
- ActiveMQTempDestinationMarshaller
 - activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller, 463
 - activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller, 474
 - activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller, 460
 - activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller, 467
 - activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller, 471
- ActiveMQTempQueue
 - activemq::commands::ActiveMQTempQueue, 478
- ActiveMQTempQueueMarshaller
 - activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller, 486
 - activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller, 498
 - activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller, 482
 - activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller, 490
 - activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller, 494
- ActiveMQTempTopic
 - activemq::commands::ActiveMQTempTopic, 502
- ActiveMQTempTopicMarshaller
 - activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller, 510
 - activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller, 522
 - activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller, 506
 - activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller, 514
 - activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller, 518
- ActiveMQTextMessage
 - activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessage, 526
 - activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller, 535
 - activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller, 531
 - activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller, 537
 - activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller, 541
- ActiveMQTopic
 - activemq::commands::ActiveMQTopic, 551
 - activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller, 558
 - activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller, 570
 - activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller, 554
 - activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller, 562
 - activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller, 566
- ActiveMQTransactionContext
 - activemq::core::ActiveMQTransactionContext, 574
- add
 - activemq::transport::failover::CloseTransportsTask, 953
 - activemq::transport::failover::FailoverTransport, 953
- decaf::util::AbstractCollection, 122
- decaf::util::AbstractQueue, 135
- decaf::util::Collection, 984
- decaf::util::List, 1867
- decaf::util::ListIterator, 1872
- decaf::util::PriorityQueue, 2416
- decaf::util::StlList, 2838, 2839
- decaf::util::StlSet, 2868
- addAll
 - decaf::util::AbstractCollection, 122
 - decaf::util::AbstractQueue, 135
 - decaf::util::List, 1867
- addAndGet
 - decaf::util::AtomicInteger, 583
- addConnection
 - activemq::state::TransactionState, 3061

- activemq::cmsutil::ResourceLifecycleManager, 2609
- addConsumer
 - activemq::state::SessionState, 2728
- addDestination
 - activemq::cmsutil::ResourceLifecycleManager, 2609
- addDispatcher
 - activemq::core::ActiveMQConnection, 205
- addHandler
 - decaf::util::logging::Logger, 1910
- additionalPredicate
 - activemq::commands::ConsumerInfo, 1222
- addMarshaller
 - activemq::wireformat::openwire::OpenWireFormat, 2300
- addMessageConsumer
 - activemq::cmsutil::ResourceLifecycleManager, 2610
- addMessageProducer
 - activemq::cmsutil::ResourceLifecycleManager, 2610
- addProducer
 - activemq::core::ActiveMQConnection, 205
 - activemq::state::SessionState, 2728
- addPropertyChangeListener
 - decaf::util::logging::LogManager, 1925
- addSession
 - activemq::cmsutil::ResourceLifecycleManager, 2610
 - activemq::state::ConnectionState, 1159
- addSynchronization
 - activemq::core::ActiveMQTransactionContext, 574
- addTask
 - activemq::threads::CompositeTaskRunner, 1018
- addTempDestination
 - activemq::state::ConnectionState, 1159
- addTransactionState
 - activemq::state::ConnectionState, 1159
- addTransportListener
 - activemq::core::ActiveMQConnection, 205
- addURI
 - activemq::transport::CompositeTransport, 1020
 - activemq::transport::failover::FailoverTransport, 1515
 - activemq::transport::failover::URIPool, 3118
- addURIs
 - activemq::transport::failover::URIPool, 3118
- adjustMinimum
 - decaf::internal::util::TimerTaskHeap, 3003
- advisory
 - activemq::commands::ActiveMQDestination, 248
- ADVISORY_PREFIX
 - activemq::commands::ActiveMQDestination, 248
- after
 - decaf::util::Date, 1379
- afterCommit
 - activemq::core::Synchronization, 2946
- afterMarshal
 - activemq::commands::BaseDataStructure, 660
- activemq::wireformat::MarshalAware, 1993
- afterMessageIsConsumed
 - activemq::core::ActiveMQConsumer, 233
- afterRollback
 - activemq::core::Synchronization, 2946
- afterUnmarshal
 - activemq::commands::BaseDataStructure, 660
 - activemq::commands::Message, 2022
 - activemq::commands::WireFormatInfo, 3155
 - activemq::wireformat::MarshalAware, 1993
- allocate
 - decaf::nio::ByteBuffer, 854
 - decaf::nio::CharBuffer, 934
 - decaf::nio::DoubleBuffer, 1462
 - decaf::nio::FloatBuffer, 1564
 - decaf::nio::IntBuffer, 1644
 - decaf::nio::LongBuffer, 1958
 - decaf::nio::ShortBuffer, 2748
- AMQ_CATCH_ALL_THROW - CMSEXCEPTION
 - CMSExceptionSupport.h, 3304
- AMQ_CATCH_EXCEPTION_CONVERT
 - activemq/exceptions/ExceptionDefines.h, 3272
- AMQ_CATCH_NOTHROW
 - activemq/exceptions/ExceptionDefines.h, 3273
- AMQ_CATCH_RETHROW
 - activemq/exceptions/ExceptionDefines.h, 3273
- AMQ_CATCHALL_NOTHROW
 - activemq/exceptions/ExceptionDefines.h, 3273
- AMQ_CATCHALL_THROW
 - activemq/exceptions/ExceptionDefines.h, 3273
- AMQCPP_API
 - activemq/util/Config.h, 3306

- ANY_CHILD
 - activemq::commands::ActiveMQDestination::DestinationFilter, 1390
- ANY_DESCENDENT
 - activemq::commands::ActiveMQDestination::DestinationFilter, 1390
- append
 - decaf::lang::Appendable, 577
 - decaf::nio::CharBuffer, 934, 935
- AprPool
 - decaf::internal::AprPool, 579
- array
 - decaf::internal::nio::ByteBuffer, 812
 - decaf::internal::nio::CharArrayBuffer, 925
 - decaf::internal::nio::DoubleArrayBuffer, 1455
 - decaf::internal::nio::FloatArrayBuffer, 1557
 - decaf::internal::nio::IntArrayBuffer, 1637
 - decaf::internal::nio::LongArrayBuffer, 1951
 - decaf::internal::nio::ShortArrayBuffer, 2741
 - decaf::nio::ByteBuffer, 854
 - decaf::nio::CharBuffer, 935
 - decaf::nio::DoubleBuffer, 1462
 - decaf::nio::FloatBuffer, 1565
 - decaf::nio::IntBuffer, 1644
 - decaf::nio::LongBuffer, 1958
 - decaf::nio::ShortBuffer, 2748
- arrayOffset
 - decaf::internal::nio::ByteBuffer, 812
 - decaf::internal::nio::CharArrayBuffer, 925
 - decaf::internal::nio::DoubleArrayBuffer, 1455
 - decaf::internal::nio::FloatArrayBuffer, 1558
 - decaf::internal::nio::IntArrayBuffer, 1637
 - decaf::internal::nio::LongArrayBuffer, 1951
 - decaf::internal::nio::ShortArrayBuffer, 2741
 - decaf::nio::ByteBuffer, 854
 - decaf::nio::CharBuffer, 936
 - decaf::nio::DoubleBuffer, 1463
 - decaf::nio::FloatBuffer, 1565
 - decaf::nio::IntBuffer, 1644
 - decaf::nio::LongBuffer, 1958
 - decaf::nio::ShortBuffer, 2749
- arrival
 - activemq::commands::Message, 2035
- asCharBuffer
 - decaf::internal::nio::ByteBuffer, 813
 - decaf::nio::ByteBuffer, 855
- asDoubleBuffer
 - decaf::internal::nio::ByteBuffer, 813
 - decaf::nio::ByteBuffer, 855
- asFloatBuffer
 - decaf::internal::nio::ByteBuffer, 813
 - decaf::nio::ByteBuffer, 855
- asIntBuffer
 - decaf::internal::nio::ByteBuffer, 814
 - decaf::nio::ByteBuffer, 856
- asLongBuffer
 - decaf::internal::nio::ByteBuffer, 814
 - decaf::nio::ByteBuffer, 856
- asReadOnlyBuffer
 - decaf::internal::nio::ByteBuffer, 814
 - decaf::internal::nio::CharArrayBuffer, 926
 - decaf::internal::nio::DoubleArrayBuffer, 1456
 - decaf::internal::nio::FloatArrayBuffer, 1558
 - decaf::internal::nio::IntArrayBuffer, 1637
 - decaf::internal::nio::LongArrayBuffer, 1951
 - decaf::internal::nio::ShortArrayBuffer, 2742
 - decaf::nio::ByteBuffer, 856
 - decaf::nio::CharBuffer, 936
 - decaf::nio::DoubleBuffer, 1463
 - decaf::nio::FloatBuffer, 1565
 - decaf::nio::IntBuffer, 1645
 - decaf::nio::LongBuffer, 1959
 - decaf::nio::ShortBuffer, 2749
- asShortBuffer
 - decaf::internal::nio::ByteBuffer, 815
 - decaf::nio::ByteBuffer, 857
- AsyncSignalReadErrorTask
 - activemq::transport::inactivity::InactivityMonitor, 1627
- AsyncWriteTask
 - activemq::transport::inactivity::InactivityMonitor, 1627
- AtomicBoolean
 - decaf::util::concurrent::atomic::AtomicBoolean, 580
- AtomicInteger
 - decaf::util::concurrent::atomic::AtomicInteger, 583
- AtomicRefCounter
 - decaf::lang::AtomicRefCounter, 588
- AtomicReference
 - decaf::util::concurrent::atomic::AtomicReference, 590
- AUTO_ACKNOWLEDGE
 - cms::Session, 2667
- available
 - decaf::internal::io::StandardInputStream, 2820
 - decaf::io::BlockingByteArrayInputStream, 668
 - decaf::io::BufferedInputStream, 749
 - decaf::io::ByteArrayInputStream, 830

- decaf::io::FilterInputStream, 1530
- decaf::io::InputStream, 1631
- decaf::net::SocketInputStream, 2798
- availablePermits
 - decaf::util::concurrent::Semaphore, 2656
- availableProcessors
 - decaf::lang::System, 2954
- await
 - decaf::util::concurrent::CountDownLatch, 1267
 - decaf::util::concurrent::locks::Condition, 1042, 1043
- awaitNanos
 - decaf::util::concurrent::locks::Condition, 1043
- awaitTermination
 - decaf::util::concurrent::ExecutorService, 1512
- awaitUninterruptibly
 - decaf::util::concurrent::locks::Condition, 1045
- awaitUntil
 - decaf::util::concurrent::locks::Condition, 1045
- back
 - decaf::util::StlQueue, 2860
- BackupTransport
 - activemq::transport::failover::BackupTransport, 592
 - activemq::transport::failover::BackupTransportPool, 596
- BackupTransportPool
 - activemq::transport::failover::BackupTransportPool, 595
- BaseCommand
 - activemq::commands::BaseCommand, 598
- BaseCommandMarshaller
 - activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller, 611
 - activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller, 631
 - activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller, 604
 - activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller, 618
 - activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller, 624
- before
 - decaf::util::Date, 1379
- beforeEnd
 - activemq::core::Synchronization, 2946
- beforeMarshal
 - activemq::commands::ActiveMQMapMessage, 275
 - activemq::commands::ActiveMQTextMessage, 526
 - activemq::commands::BaseDataStructure, 660
 - activemq::commands::Message, 2022
 - activemq::commands::WireFormatInfo, 3155
 - activemq::wireformat::MarshalAware, 1994
- beforeMessageIsConsumed
 - activemq::core::ActiveMQConsumer, 233
- beforeUnmarshal
 - activemq::commands::BaseDataStructure, 660
 - activemq::wireformat::MarshalAware, 1994
- BEGIN
 - activemq::wireformat::stomp::StompCommandConstants, 2873
- begin
 - activemq::core::ActiveMQTransactionContext, 574
- BIG_STRING_TYPE
 - activemq::util::PrimitiveValueNode, 2402
- BINARY_MIME_TYPE
 - activemq::commands::ActiveMQBlobMessage, 146
- bind
 - decaf::net::ServerSocket, 2662, 2663
- BindException
 - decaf::net::BindException, 664, 665
- bitCount
 - decaf::lang::Integer, 1656
 - decaf::lang::Long, 1937
- BlockingByteArrayInputStream
 - decaf::io::BlockingByteArrayInputStream, 668
- Boolean
 - decaf::lang::Boolean, 675
- BOOLEAN_TYPE
 - activemq::util::PrimitiveValueNode, 2401
- BooleanExpression
 - activemq::wireformat::openwire::utils::BooleanExpression, 679
- BooleanStream
 - activemq::wireformat::openwire::utils::BooleanStream, 682
- booleanValue
 - decaf::lang::Boolean, 675
- boolValue
 - activemq::util::PrimitiveValueNode::PrimitiveValue, 2396
- branchQualifier

- activemq::commands::XATransactionId, 3197
- BrokenBarrierException
 - decaf::util::concurrent::BrokenBarrierException, 684, 685
- BrokerError
 - activemq::commands::BrokerError, 687
- BrokerException
 - activemq::exceptions::BrokerException, 690
- BrokerId
 - activemq::commands::BrokerId, 692
- brokerId
 - activemq::commands::BrokerInfo, 721
- BrokerIdMarshaller
 - activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller, 699
 - activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller, 711
 - activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller, 695
 - activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller, 703
 - activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller, 707
- BrokerInfo
 - activemq::commands::BrokerInfo, 715
- BrokerInfoMarshaller
 - activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller, 727
 - activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller, 739
 - activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller, 723
 - activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller, 731
 - activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller, 735
- brokerInTime
 - activemq::commands::Message, 2035
- brokerMasterConnector
 - activemq::commands::ConnectionInfo, 1134
- brokerName
 - activemq::commands::BrokerInfo, 721
 - activemq::commands::DiscoveryEvent, 1420
- brokerOutTime
 - activemq::commands::Message, 2035
- brokerPath
 - activemq::commands::ConnectionInfo, 1134
 - activemq::commands::ConsumerInfo, 1222
- activemq::commands::DestinationInfo, 1395
- activemq::commands::Message, 2035
- activemq::commands::ProducerInfo, 2474
- brokerSequenceId
 - activemq::commands::MessageId, 2146
- brokerUploadUrl
 - activemq::commands::BrokerInfo, 721
- brokerURL
 - activemq::commands::BrokerInfo, 721
- browser
 - activemq::commands::ConsumerInfo, 1222
- Buffer
 - decaf::nio::Buffer, 744
- buffer
 - decaf::io::DataOutputStream, 1307
 - BufferedInputStream
 - decaf::io::BufferedInputStream, 748, 749
 - BufferedOutputStream
 - decaf::io::BufferedOutputStream, 753
 - BufferedSocket
 - decaf::net::BufferedSocket, 756
 - BufferOverflowException
 - decaf::nio::BufferOverflowException, 772, 773
 - BufferUnderflowException
 - decaf::nio::BufferUnderflowException, 774, 775
- buildIncomingCommands
 - activemq::transport::mock::ResponseBuilder, 2615
- activemq::wireformat::openwire::OpenWireResponseBuilder, 2314
- buildMessage
 - decaf::lang::Exception, 1479
- buildResponse
 - activemq::transport::mock::ResponseBuilder, 2613
- activemq::wireformat::openwire::OpenWireResponseBuilder, 2314
- Byte
 - decaf::lang::Byte, 778
- BYTE_ARRAY_TYPE
 - activemq::util::PrimitiveValueNode, 2402
- BYTE_TYPE
 - activemq::util::PrimitiveValueNode, 2402
- ByteArrayAdapter
 - decaf::internal::util::ByteArrayAdapter, 789–791
- ByteArrayBuffer
 - decaf::internal::nio::ByteArrayBuffer, 811
- ByteArrayInputStream
 - decaf::io::ByteArrayInputStream, 830
- ByteArrayOutputStream

- decaf::io::ByteArrayOutputStream, 837
- ByteArrayPerspective
 - decaf::internal::nio::ByteArrayPerspective, 845–847
- byteArrayValue
 - activemq::util::PrimitiveValueNode::PrimitiveValue, 2396
- ByteBuffer
 - decaf::nio::ByteBuffer, 853
- BYTES
 - activemq::wireformat::stomp::StompCommandConstants, 2873
- byteValue
 - activemq::util::PrimitiveValueNode::PrimitiveValue, 2396
 - decaf::lang::Byte, 778
 - decaf::lang::Character, 917
 - decaf::lang::Double, 1444
 - decaf::lang::Float, 1546
 - decaf::lang::Integer, 1656
 - decaf::lang::Long, 1937
 - decaf::lang::Number, 2283
 - decaf::lang::Short, 2732
- CachedConsumer
 - activemq::cmsutil::CachedConsumer, 889
- CachedProducer
 - activemq::cmsutil::CachedProducer, 892
- call
 - decaf::util::concurrent::Callable, 898
- cancel
 - decaf::util::concurrent::Future, 1598
 - decaf::util::Timer, 2992
 - decaf::util::TimerTask, 3001
- CancellationException
 - decaf::util::concurrent::CancellationException, 899, 900
- capacity
 - decaf::nio::Buffer, 744
- cause
 - decaf::lang::Exception, 1482
- ceil
 - decaf::lang::Math, 2002
- CertificateEncodingException
 - decaf::security::cert::CertificateEncodingException, 905
- CertificateException
 - decaf::security::cert::CertificateException, 907
- CertificateExpiredException
 - decaf::security::cert::CertificateExpiredException, 909
- CertificateNot Yet ValidException
 - decaf::security::cert::CertificateNotYetValidException, 911
- CertificateParsingException
 - decaf::security::cert::CertificateParsingException, 913
- CHAR
 - CHAR_ TYPE
- activemq::util::PrimitiveValueNode, 2402
- Character
 - decaf::lang::Character, 916
- CharArrayBuffer
 - decaf::internal::nio::CharArrayBuffer, 924
- charAt
 - decaf::lang::CharSequence, 947
- decaf::nio::CharBuffer, 936
- CharBuffer
 - decaf::nio::CharBuffer, 933
- charValue
 - activemq::util::PrimitiveValueNode::PrimitiveValue, 2396
- checkConnectionFactory
 - activemq::cmsutil::CmsAccessor, 955
- checkDestinationResolver
 - activemq::cmsutil::CmsDestinationAccessor, 959
- checkMapIsUnmarshalled
 - activemq::commands::ActiveMQMapMessage, 275
- checkResult
 - decaf::net::TcpSocket, 2961
- checkShutdown
 - activemq::state::ConnectionState, 1159
 - activemq::state::SessionState, 2728
 - activemq::state::TransactionState, 3061
- checkValidity
 - decaf::security::cert::X509Certificate, 3190
 - decaf::security_ -
 - provider::unix::openssl::OpenSSLX509Certificate, 2291
- ClassCastException
 - decaf::lang::exceptions::ClassCastException, 949, 950
- ClassName
 - activemq::commands::BrokerError::StackTraceElement, 2812
- cleanup
 - decaf::internal::AprPool, 579
- clear
 - activemq::core::ActiveMQSessionExecutor, 420
 - activemq::core::MessageDispatchChannel, 2090
 - activemq::util::ActiveMQProperties, 376
 - activemq::util::PrimitiveValueNode, 2405

- activemq::wireformat::openwire::utils::BooleanStream, 682
- cms::CMSProperties, 966
- decaf::internal::util::ByteArrayAdapter, 791
- decaf::nio::Buffer, 744
- decaf::util::AbstractCollection, 123
- decaf::util::AbstractQueue, 136
- decaf::util::Collection, 985
- decaf::util::concurrent::ConcurrentStlMap, 1029
- decaf::util::concurrent::SynchronousQueue, 2949
- decaf::util::Map, 1972
- decaf::util::PriorityQueue, 2416
- decaf::util::Properties, 2496
- decaf::util::StlList, 2840
- decaf::util::StlMap, 2850
- decaf::util::StlQueue, 2860
- decaf::util::StlSet, 2869
- clearBody
 - activemq::commands::ActiveMQBytesMessage, 170
 - activemq::commands::ActiveMQMapMessage, 275
 - activemq::commands::ActiveMQMessageTemplate, 331
 - activemq::commands::ActiveMQStreamMessage, 425
 - activemq::commands::ActiveMQTextMessage, 526
 - cms::Message, 2040
- clearMessagesInProgress
 - activemq::core::ActiveMQConsumer, 233
 - activemq::core::ActiveMQSession, 406
 - activemq::core::ActiveMQSessionExecutor, 420
- clearProperties
 - activemq::commands::ActiveMQMessageTemplate, 331
 - cms::Message, 2041
- CLIENT_ACKNOWLEDGE
 - cms::Session, 2667
- clientId
 - activemq::commands::ConnectionInfo, 1134
 - activemq::commands::JournalTopicAck, 1746
 - activemq::commands::RemoveSubscriptionInfo, 2564
 - activemq::commands::SubscriptionInfo, 2911
- clientMaster
 - activemq::commands::ConnectionInfo, 1134
 - clockSequence
 - decaf::util::UUID, 3143
 - clone
 - activemq::commands::ActiveMQBlobMessage, 143
 - activemq::commands::ActiveMQBytesMessage, 170
 - activemq::commands::ActiveMQMapMessage, 275
 - activemq::commands::ActiveMQMessage, 305
 - activemq::commands::ActiveMQObjectMessage, 345
 - activemq::commands::ActiveMQQueue, 380
 - activemq::commands::ActiveMQStreamMessage, 425
 - activemq::commands::ActiveMQTempQueue, 478
 - activemq::commands::ActiveMQTempTopic, 502
 - activemq::commands::ActiveMQTextMessage, 527
 - activemq::commands::ActiveMQTopic, 551
 - activemq::exceptions::ActiveMQException, 271
 - activemq::exceptions::BrokerException, 691
 - activemq::util::ActiveMQProperties, 376
 - activemq::wireformat::stomp::StompFrame, 2875
 - cms::BytesMessage, 877
 - cms::CMSProperties, 966
 - cms::Destination, 1389
 - cms::Message, 2041
 - decaf::io::EOFException, 1476
 - decaf::io::InterruptedIOException, 1695
 - decaf::io::IOException, 1709
 - decaf::io::UnsupportedEncodingException, 3092
 - decaf::io::UTFDataFormatException, 3141
 - decaf::lang::Exception, 1479
 - decaf::lang::exceptions::ClassCastException, 950
 - decaf::lang::exceptions::IllegalArgumentException, 1615
 - decaf::lang::exceptions::IllegalMonitorStateException, 1618
 - decaf::lang::exceptions::IllegalStateException, 1620
 - decaf::lang::exceptions::IllegalThreadStateException, 1624

- decaf::lang::exceptions::IndexOutOfBoundsException, 1629
- decaf::lang::exceptions::InterruptedException, 1693
- decaf::lang::exceptions::InvalidStateException, 1706
- decaf::lang::exceptions::NoSuchElementException, 2276
- decaf::lang::exceptions::NullPointerException, 2281
- decaf::lang::exceptions::NumberFormatException, 2286
- decaf::lang::exceptions::RuntimeException, 2646
- decaf::lang::exceptions::UnsupportedOperationException, 3096
- decaf::lang::Throwable, 2984
- decaf::net::BindException, 665
- decaf::net::ConnectException, 1051
- decaf::net::HttpRetryException, 1613
- decaf::net::MalformedURLException, 1970
- decaf::net::NoRouteToHostException, 2271
- decaf::net::PortUnreachableException, 2370
- decaf::net::ProtocolException, 2506
- decaf::net::SocketException, 2794
- decaf::net::SocketTimeoutException, 2811
- decaf::net::UnknownHostException, 3087
- decaf::net::UnknownServiceException, 3089
- decaf::net::URISyntaxException, 3125
- decaf::nio::BufferOverflowException, 773
- decaf::nio::BufferUnderflowException, 776
- decaf::nio::InvalidMarkException, 1703
- decaf::nio::ReadOnlyBufferException, 2522
- decaf::security::cert::CertificateEncodingException, 906
- decaf::security::cert::CertificateException, 908
- decaf::security::cert::CertificateExpiredException, 910
- decaf::security::cert::CertificateNotYetValidException, 912
- decaf::security::cert::CertificateParsingException, 914
- decaf::security::GeneralSecurityException, 1604
- decaf::security::InvalidKeyException, 1700
- decaf::security::KeyException, 1839
- decaf::security::NoSuchAlgorithmExceptionException, 2274
- decaf::security::NoSuchProviderException, 2279
- decaf::security::SignatureException, 2781
- decaf::util::concurrent::BrokenBarrierException, 685
- decaf::util::concurrent::CancellationException, 900
- decaf::util::concurrent::ExecutionException, 1509
- decaf::util::concurrent::RejectedExecutionException, 2535
- decaf::util::concurrent::TimeoutException, 2989
- decaf::util::Properties, 2496
- cloneDataStructure
- activemq::commands::ActiveMQBlobMessage, 143
- activemq::commands::ActiveMQBytesMessage, 170
- activemq::commands::ActiveMQDestination, 242
- activemq::commands::ActiveMQMapMessage, 275
- activemq::commands::ActiveMQMessage, 305
- activemq::commands::ActiveMQObjectMessage, 345
- activemq::commands::ActiveMQQueue, 380
- activemq::commands::ActiveMQStreamMessage, 425
- activemq::commands::ActiveMQTempDestination, 457
- activemq::commands::ActiveMQTempQueue, 478
- activemq::commands::ActiveMQTempTopic, 502
- activemq::commands::ActiveMQTextMessage, 527
- activemq::commands::ActiveMQTopic, 551
- activemq::commands::BooleanExpression, 679
- activemq::commands::BrokerError, 687
- activemq::commands::BrokerId, 692
- activemq::commands::BrokerInfo, 715
- activemq::commands::ConnectionControl, 1057
- activemq::commands::ConnectionError, 1081
- activemq::commands::ConnectionId, 1107
- activemq::commands::ConnectionInfo, 1130
- activemq::commands::ConsumerControl, 1167
- activemq::commands::ConsumerId, 1192
- activemq::commands::ConsumerInfo, 1217

- activemq::commands::ControlCommand, 1244
- activemq::commands::DataArrayResponse, 1269
- activemq::commands::DataResponse, 1308
- activemq::commands::DataStructure, 1373
- activemq::commands::DestinationInfo, 1392
- activemq::commands::DiscoveryEvent, 1418
- activemq::commands::ExceptionResponse, 1485
- activemq::commands::FlushCommand, 1574
- activemq::commands::IntegerResponse, 1668
- activemq::commands::JournalQueueAck, 1719
- activemq::commands::JournalTopicAck, 1743
- activemq::commands::JournalTrace, 1767
- activemq::commands::JournalTransaction, 1791
- activemq::commands::KeepAliveInfo, 1813
- activemq::commands::LastPartialCommand, 1841
- activemq::commands::LocalTransactionId, 1876
- activemq::commands::Message, 2022
- activemq::commands::MessageAck, 2056
- activemq::commands::MessageDispatch, 2085
- activemq::commands::MessageDispatchNotification, 2117
- activemq::commands::MessageId, 2143
- activemq::commands::MessagePull, 2204
- activemq::commands::NetworkBridgeFilter, 2247
- activemq::commands::PartialCommand, 2320
- activemq::commands::ProducerAck, 2421
- activemq::commands::ProducerId, 2448
- activemq::commands::ProducerInfo, 2471
- activemq::commands::RemoveInfo, 2538
- activemq::commands::RemoveSubscriptionInfo, 2561
- activemq::commands::ReplayCommand, 2585
- activemq::commands::Response, 2612
- activemq::commands::SessionId, 2679
- activemq::commands::SessionInfo, 2703
- activemq::commands::ShutdownInfo, 2758
- activemq::commands::SubscriptionInfo, 2908
- activemq::commands::TransactionId, 3017
- activemq::commands::TransactionInfo, 3038
- activemq::commands::WireFormatInfo, 3155
- activemq::commands::XATransactionId, 3194
- close
 - activemq::cmsutil::CachedConsumer, 889
 - activemq::cmsutil::CachedProducer, 892
 - activemq::cmsutil::PooledSession, 2354
 - activemq::commands::ActiveMQTempDestination, 457
 - activemq::commands::ConnectionControl, 1059
 - activemq::commands::ConsumerControl, 1170
 - activemq::core::ActiveMQConnection, 205
 - activemq::core::ActiveMQConsumer, 233
 - activemq::core::ActiveMQProducer, 369
 - activemq::core::ActiveMQSession, 407
 - activemq::core::ActiveMQSessionExecutor, 420
 - activemq::core::MessageDispatchChannel, 2090
 - activemq::transport::correlator::ResponseCorrelator, 2617
 - activemq::transport::failover::FailoverTransport, 1515
 - activemq::transport::inactivity::InactivityMonitor, 1625
 - activemq::transport::IOTransport, 1711
 - activemq::transport::mock::MockTransport, 2229
 - activemq::transport::tcp::TcpTransport, 2968
 - activemq::transport::TransportFilter, 3075
 - activemq::wireformat::openwire::OpenWireFormatNegotiator, 2311
- cms::Closeable, 952
- cms::Connection, 1053
- cms::Session, 2667
- decaf::internal::io::StandardErrorOutputStream, 2814
- decaf::internal::io::StandardInputStream, 2820
- decaf::internal::io::StandardOutputStream, 2826
- decaf::io::BlockingByteArrayInputStream, 668
- decaf::io::BufferedInputStream, 749
- decaf::io::BufferedOutputStream, 753
- decaf::io::ByteArrayInputStream, 830
- decaf::io::ByteArrayOutputStream, 838

- decaf::io::Closeable, 951
- decaf::io::FilterInputStream, 1530
- decaf::io::FilterOutputStream, 1539
- decaf::net::BufferedSocket, 757
- decaf::net::ServerSocket, 2663
- decaf::net::SocketInputStream, 2798
- decaf::net::SocketOutputStream, 2805
- decaf::net::TcpSocket, 2961
- decaf::util::logging::StreamHandler, 2889
- closed
 - decaf::io::FilterInputStream, 1536
 - decaf::io::FilterOutputStream, 1543
- CloseTransportsTask
 - activemq::transport::failover::CloseTransportsTask, 953
- cluster
 - activemq::commands::Message, 2035
- cms, 99
- cms/Config.h
 - CMS_API, 3306
- cms::BytesMessage, 874
 - ~BytesMessage, 877
 - clone, 877
 - getBodyBytes, 877
 - getBodyLength, 878
 - readBoolean, 878
 - readByte, 878
 - readBytes, 879
 - readChar, 880
 - readDouble, 880
 - readFloat, 880
 - readInt, 881
 - readLong, 881
 - readShort, 881
 - readString, 882
 - readUnsignedShort, 882
 - readUTF, 882
 - reset, 883
 - setBodyBytes, 883
 - writeBoolean, 883
 - writeByte, 884
 - writeBytes, 884
 - writeChar, 885
 - writeDouble, 885
 - writeFloat, 885
 - writeInt, 886
 - writeLong, 886
 - writeShort, 886
 - writeString, 887
 - writeUnsignedShort, 887
 - writeUTF, 887
- cms::Closeable, 951
 - ~Closeable, 952
 - close, 952
- cms::CMSException, 960
 - ~CMSException, 962
 - CMSException, 962
 - getCause, 962
 - getMessage, 962
 - getStackTrace, 962
 - getStackTraceString, 962
 - printStackTrace, 963
 - setMark, 963
 - what, 963
- cms::CMSProperties, 965
 - ~CMSProperties, 966
 - clear, 966
 - clone, 966
 - copy, 966
 - getProperty, 966
 - hasProperty, 967
 - isEmpty, 967
 - remove, 967
 - setProperty, 967
 - toArray, 968
 - toString, 968
- cms::CMSSecurityException, 968
 - ~CMSSecurityException, 969
 - CMSSecurityException, 969
- cms::Connection, 1052
 - ~Connection, 1053
 - close, 1053
 - createSession, 1054
 - getClientID, 1054
 - getExceptionListener, 1054
 - getMetaData, 1054
 - setExceptionListener, 1055
- cms::ConnectionFactory, 1103
 - ~ConnectionFactory, 1104
 - createCMSConnectionFactory, 1104
 - createConnection, 1104, 1105
- cms::ConnectionMetaData, 1154
 - ~ConnectionMetaData, 1155
 - getCMSMajorVersion, 1155
 - getCMSMinorVersion, 1156
 - getCMSProviderName, 1156
 - getCMSVersion, 1156
 - getCMSXPropertyNames, 1156
 - getProviderMajorVersion, 1157
 - getProviderMinorVersion, 1157
 - getProviderVersion, 1157
- cms::DeliveryMode, 1386
 - ~DeliveryMode, 1387
 - DELIVERY_MODE, 1387
 - NON_PERSISTENT, 1387
 - PERSISTENT, 1387
- cms::Destination, 1387
 - ~Destination, 1389

- clone, 1389
- copy, 1389
- DestinationType, 1388
- getCMSProperties, 1389
- getDestinationType, 1389
- QUEUE, 1388
- TEMPORARY_QUEUE, 1388
- TEMPORARY_TOPIC, 1388
- TOPIC, 1388
- cms::ExceptionListener, 1483
 - ~ExceptionListener, 1483
 - onException, 1483
- cms::IllegalStateException, 1621
 - ~IllegalStateException, 1621
 - IllegalStateException, 1621
- cms::InvalidClientIdException, 1696
 - ~InvalidClientIdException, 1697
 - InvalidClientIdException, 1697
- cms::InvalidDestinationException, 1697
 - ~InvalidDestinationException, 1698
 - InvalidDestinationException, 1698
- cms::InvalidSelectorException, 1703
 - ~InvalidSelectorException, 1704
 - InvalidSelectorException, 1704
- cms::MapMessage, 1982
 - ~MapMessage, 1985
 - getBoolean, 1985
 - getByte, 1985
 - getBytes, 1985
 - getChar, 1986
 - getDouble, 1986
 - getFloat, 1986
 - getInt, 1986
 - getLong, 1987
 - getMapNames, 1987
 - getShort, 1987
 - getString, 1988
 - itemExists, 1988
 - setBoolean, 1988
 - setByte, 1988
 - setBytes, 1989
 - setChar, 1989
 - setDouble, 1989
 - setFloat, 1990
 - setInt, 1990
 - setLong, 1990
 - setShort, 1991
 - setString, 1991
- cms::Message, 2036
 - ~Message, 2040
 - acknowledge, 2040
 - clearBody, 2040
 - clearProperties, 2041
 - clone, 2041
 - getBooleanProperty, 2041
 - getByteProperty, 2041
 - getCMSCorrelationID, 2042
 - getCMSDeliveryMode, 2042
 - getCMSDestination, 2042
 - getCMSExpiration, 2043
 - getCMSMessageID, 2043
 - getCMSPriority, 2044
 - getCMSRedelivered, 2044
 - getCMSReplyTo, 2044
 - getCMSTimestamp, 2045
 - getCMSType, 2045
 - getDoubleProperty, 2045
 - getFloatProperty, 2046
 - getIntProperty, 2046
 - getLongProperty, 2046
 - getPropertyNames, 2047
 - getShortProperty, 2047
 - getStringProperty, 2047
 - propertyExists, 2048
 - setBooleanProperty, 2048
 - setByteProperty, 2048
 - setCMSCorrelationID, 2049
 - setCMSDeliveryMode, 2049
 - setCMSDestination, 2050
 - setCMSExpiration, 2050
 - setCMSMessageID, 2050
 - setCMSPriority, 2051
 - setCMSRedelivered, 2051
 - setCMSReplyTo, 2051
 - setCMSTimestamp, 2052
 - setCMSType, 2052
 - setDoubleProperty, 2053
 - setFloatProperty, 2053
 - setIntProperty, 2053
 - setLongProperty, 2053
 - setShortProperty, 2054
 - setStringProperty, 2054
- cms::MessageConsumer, 2080
 - ~MessageConsumer, 2081
 - getMessageListener, 2081
 - getMessageSelector, 2081
 - receive, 2081, 2082
 - receiveNoWait, 2082
 - setMessageListener, 2082
- cms::MessageEOFException, 2140
 - ~MessageEOFException, 2141
 - MessageEOFException, 2141
- cms::MessageFormatException, 2141
 - ~MessageFormatException, 2142
 - MessageFormatException, 2142
- cms::MessageListener, 2165
 - ~MessageListener, 2166
 - onMessage, 2166

- cms::MessageNotReadableException, 2187
 - ~MessageNotReadableException, 2188
 - MessageNotReadableException, 2188
- cms::MessageNotWriteableException, 2188
 - ~MessageNotWriteableException, 2189
 - MessageNotWriteableException, 2189
- cms::MessageProducer, 2189
 - ~MessageProducer, 2191
 - getDeliveryMode, 2191
 - getDisableMessageID, 2191
 - getDisableMessageTimeStamp, 2191
 - getPriority, 2192
 - getTimeToLive, 2192
 - send, 2192–2194
 - setDeliveryMode, 2194
 - setDisableMessageID, 2195
 - setDisableMessageTimeStamp, 2195
 - setPriority, 2195
 - setTimeToLive, 2196
- cms::ObjectMessage, 2287
 - ~ObjectMessage, 2287
- cms::Queue, 2510
 - ~Queue, 2511
 - getQueueName, 2511
- cms::QueueBrowser, 2511
 - ~QueueBrowser, 2512
 - getEnumeration, 2512
 - getMessageSelector, 2512
 - getQueue, 2512
- cms::Session, 2663
 - ~Session, 2667
 - AcknowledgeMode, 2667
 - AUTO_ACKNOWLEDGE, 2667
 - CLIENT_ACKNOWLEDGE, 2667
 - close, 2667
 - commit, 2667
 - createBrowser, 2668
 - createBytesMessage, 2668, 2669
 - createConsumer, 2669, 2670
 - createDurableConsumer, 2670
 - createMapMessage, 2671
 - createMessage, 2671
 - createProducer, 2672
 - createQueue, 2672
 - createStreamMessage, 2672
 - createTemporaryQueue, 2673
 - createTemporaryTopic, 2673
 - createTextMessage, 2673
 - createTopic, 2674
 - DUPS_OK_ACKNOWLEDGE, 2667
 - getAcknowledgeMode, 2674
 - INDIVIDUAL_ACKNOWLEDGE, 2667
 - isTransacted, 2674
 - recover, 2675
 - rollback, 2675
 - SESSION_TRANSACTED, 2667
 - unsubscribe, 2675
- cms::Startable, 2831
 - ~Startable, 2831
 - start, 2831
- cms::Stoppable, 2887
 - ~Stoppable, 2888
 - stop, 2888
- cms::StreamMessage, 2892
 - ~StreamMessage, 2895
 - readBoolean, 2895
 - readByte, 2895
 - readBytes, 2896
 - readChar, 2897
 - readDouble, 2897
 - readFloat, 2898
 - readInt, 2898
 - readLong, 2898
 - readShort, 2899
 - readString, 2899
 - readUnsignedShort, 2900
 - writeBoolean, 2900
 - writeByte, 2900
 - writeBytes, 2901
 - writeChar, 2901
 - writeDouble, 2902
 - writeFloat, 2902
 - writeInt, 2902
 - writeLong, 2903
 - writeShort, 2903
 - writeString, 2903
 - writeUnsignedShort, 2903
- cms::TemporaryQueue, 2971
 - ~TemporaryQueue, 2972
 - destroy, 2972
 - getQueueName, 2972
- cms::TemporaryTopic, 2973
 - ~TemporaryTopic, 2973
 - destroy, 2973
 - getTopicName, 2973
- cms::TextMessage, 2974
 - ~TextMessage, 2975
 - getText, 2975
 - setText, 2975
- cms::Topic, 3014
 - ~Topic, 3014
 - getTopicName, 3014
- cms::UnsupportedOperationException, 3093
 - ~UnsupportedOperationException, 3093
 - UnsupportedOperationException, 3093
- CMS_API
 - cms/Config.h, 3306
- CmsAccessor

- activemq::cmsutil::CmsAccessor, 955
- CmsDestinationAccessor
 - activemq::cmsutil::CmsDestinationAccessor, 959
- CMSException
 - cms::CMSException, 962
- CMSExceptionSupport.h
 - AMQ_CATCH_ALL_THROW - CMSEXCEPTION, 3304
- CMSecurityException
 - cms::CMSecurityException, 969
- CmsTemplate
 - activemq::cmsutil::CmsTemplate, 973
- command
 - activemq::commands::ControlCommand, 1246
- commandId
 - activemq::commands::PartialCommand, 2322
- COMMIT
 - activemq::wireformat::stomp::StompCommandConstants, 2873
- commit
 - activemq::cmsutil::PooledSession, 2354
 - activemq::core::ActiveMQConsumer, 233
 - activemq::core::ActiveMQSession, 407
 - activemq::core::ActiveMQTransactionContext, 575
 - cms::Session, 2667
- compact
 - decaf::internal::nio::ByteBuffer, 815
 - decaf::internal::nio::CharArrayBuffer, 926
 - decaf::internal::nio::DoubleArrayBuffer, 1456
 - decaf::internal::nio::FloatArrayBuffer, 1558
 - decaf::internal::nio::IntArrayBuffer, 1638
 - decaf::internal::nio::LongArrayBuffer, 1952
 - decaf::internal::nio::ShortArrayBuffer, 2742
 - decaf::nio::ByteBuffer, 857
 - decaf::nio::CharBuffer, 937
 - decaf::nio::DoubleBuffer, 1463
 - decaf::nio::FloatBuffer, 1566
 - decaf::nio::IntBuffer, 1645
 - decaf::nio::LongBuffer, 1959
 - decaf::nio::ShortBuffer, 2749
- COMPARATOR
 - activemq::commands::BrokerId, 692
 - activemq::commands::ConnectionId, 1107
 - activemq::commands::ConsumerId, 1192
 - activemq::commands::LocalTransactionId, 1876
 - activemq::commands::MessageId, 2143
 - activemq::commands::ProducerId, 2447
- activemq::commands::SessionId, 2679
- activemq::commands::TransactionId, 3017
- activemq::commands::XATransactionId, 3194
- comparator
 - decaf::util::PriorityQueue, 2417
- compare
 - decaf::lang::Double, 1444
 - decaf::lang::Float, 1547
 - decaf::lang::PointerComparator, 2351
 - decaf::util::Comparator, 1012
 - decaf::util::comparators::Less, 1863
- compareAndSet
 - decaf::util::concurrent::atomic::AtomicBoolean, 580
 - decaf::util::concurrent::atomic::AtomicInteger, 584
 - decaf::util::concurrent::atomic::AtomicReference, 590
- compareTo
 - activemq::commands::BrokerId, 692
 - activemq::commands::ConnectionId, 1107
 - activemq::commands::ConsumerId, 1192
 - activemq::commands::LocalTransactionId, 1876
 - activemq::commands::MessageId, 2143
 - activemq::commands::ProducerId, 2448
 - activemq::commands::SessionId, 2679
 - activemq::commands::TransactionId, 3017
 - activemq::commands::XATransactionId, 3194
 - decaf::lang::Boolean, 675
 - decaf::lang::Byte, 778, 779
 - decaf::lang::Character, 917
 - decaf::lang::Comparable, 1010
 - decaf::lang::Double, 1444
 - decaf::lang::Float, 1547
 - decaf::lang::Integer, 1656
 - decaf::lang::Long, 1937, 1938
 - decaf::lang::Short, 2732
 - decaf::net::URI, 3100
 - decaf::nio::ByteBuffer, 857
 - decaf::nio::CharBuffer, 937
 - decaf::nio::DoubleBuffer, 1464
 - decaf::nio::FloatBuffer, 1566
 - decaf::nio::IntBuffer, 1646
 - decaf::nio::LongBuffer, 1960
 - decaf::nio::ShortBuffer, 2750
 - decaf::util::concurrent::TimeUnit, 3007
 - decaf::util::Date, 1379
 - decaf::util::UUID, 3144
- COMPOSITE_SEPARATOR
 - activemq::commands::ActiveMQDestination, 248

- CompositeData
 - activemq::util::CompositeData, 1015
- CompositeTaskRunner
 - activemq::threads::CompositeTaskRunner, 1018
- compressed
 - activemq::commands::Message, 2035
- Concurrent.h
 - synchronized, 3620
 - WAIT_INFINITE, 3620
- ConcurrentStlMap
 - decaf::util::concurrent::ConcurrentStlMap, 1029
- condition
 - decaf::util::concurrent::ConditionHandle, 1047
- ConditionHandle
 - decaf::util::concurrent::ConditionHandle, 1047
- configure
 - activemq::wireformat::openwire::marshal::v1::MarshallerFactory, 1997
 - activemq::wireformat::openwire::marshal::v2::MarshallerFactory, 1996
 - activemq::wireformat::openwire::marshal::v3::MarshallerFactory, 1998
 - activemq::wireformat::openwire::marshal::v4::MarshallerFactory, 1996
 - activemq::wireformat::openwire::marshal::v5::MarshallerFactory, 1999
- CONNECT
 - activemq::wireformat::stomp::StompCommandConstants, 2873
- connect
 - decaf::net::BufferedSocket, 757
 - decaf::net::Socket, 2787
 - decaf::net::TcpSocket, 2961, 2962
- CONNECTED
 - activemq::wireformat::stomp::StompCommandConstants, 2873
- ConnectException
 - decaf::net::ConnectException, 1050, 1051
- connection
 - activemq::commands::ActiveMQTempDestination, 458
- CONNECTION_ADVISORY_PREFIX
 - activemq::commands::ActiveMQDestination, 248
- CONNECTION_ALWAYS_SYNC_SEND
 - activemq::core::ActiveMQConstants, 229
- CONNECTION_CLOSE_TIMEOUT
 - activemq::core::ActiveMQConstants, 229
- CONNECTION_PRODUCER_WINDOW_SIZE
 - activemq::core::ActiveMQConstants, 229
- CONNECTION_SEND_TIMEOUT
 - activemq::core::ActiveMQConstants, 229
- CONNECTION_USE_ASYNC_SEND
 - activemq::core::ActiveMQConstants, 229
- ConnectionControl
 - activemq::commands::ConnectionControl, 1057
- ConnectionControlMarshaller
 - activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller, 1065
 - activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller, 1077
 - activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller, 1061
 - activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller, 1069
 - activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller, 1073
- ConnectionError
 - activemq::commands::ConnectionError, 1088
- ConnectionFactory
 - activemq::wireformat::openwire::marshal::v1::ConnectionFactory, 1088
 - activemq::wireformat::openwire::marshal::v2::ConnectionFactory, 1100
 - activemq::wireformat::openwire::marshal::v3::ConnectionFactory, 1084
 - activemq::wireformat::openwire::marshal::v4::ConnectionFactory, 1092
 - activemq::wireformat::openwire::marshal::v5::ConnectionFactory, 1096
- ConnectionId
 - activemq::commands::ConnectionId, 1107
- connectionId
 - activemq::commands::BrokerInfo, 721
 - activemq::commands::ConnectionError, 1083
 - activemq::commands::ConnectionInfo, 1134
 - activemq::commands::ConsumerId, 1194
 - activemq::commands::DestinationInfo, 1395
 - activemq::commands::LocalTransactionId, 1878
 - activemq::commands::ProducerId, 2449
 - activemq::commands::RemoveSubscriptionInfo, 2564
 - activemq::commands::SessionId, 2681
 - activemq::commands::TransactionInfo, 3040
- ConnectionIdMarshaller
 - activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller, 1114

- activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller, 1126
- activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller, 1110
- activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller, 1118
- activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller, 1122
- ConnectionInfo
 - activemq::commands::ConnectionInfo, 1130
- ConnectionInfoMarshaller
 - activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller, 1144
 - activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller, 1152
 - activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller, 1136
 - activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller, 1140
 - activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller, 1148
- ConnectionState
 - activemq::state::ConnectionState, 1159
- ConnectionStateTracker
 - activemq::state::ConnectionStateTracker, 1162
- CONSUMER_ADVISORY_PREFIX
 - activemq::commands::ActiveMQDestination, 248
- CONSUMER_DISPATCHASYNC
 - activemq::core::ActiveMQConstants, 229
- CONSUMER_EXCLUSIVE
 - activemq::core::ActiveMQConstants, 229
- CONSUMER_NOLOCAL
 - activemq::core::ActiveMQConstants, 229
- CONSUMER_PREFETCHSIZE
 - activemq::core::ActiveMQConstants, 229
- CONSUMER_PRIORITY
 - activemq::core::ActiveMQConstants, 229
- CONSUMER_RETROACTIVE
 - activemq::core::ActiveMQConstants, 229
- CONSUMER_SELECTOR
 - activemq::core::ActiveMQConstants, 229
- ConsumerControl
 - activemq::commands::ConsumerControl, 1167
- ConsumerControlMarshaller
 - activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller, 1180
 - activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller, 1188
 - activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller, 1172
- ConsumerIdMarshaller
 - activemq::commands::ConsumerId, 1192
 - activemq::commands::ConsumerControl, 1170
 - activemq::commands::ConsumerInfo, 1222
 - activemq::commands::MessageAck, 2060
 - activemq::commands::MessageDispatch, 2036
 - activemq::commands::MessageDispatchNotification, 2120
 - activemq::commands::MessagePull, 2207
- ConsumerIdMarshaller
 - activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller, 1204
 - activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller, 1212
 - activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller, 1196
 - activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller, 1200
 - activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller, 1208
- ConsumerInfo
 - activemq::commands::ConsumerInfo, 1217
- ConsumerInfoMarshaller
 - activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller, 1228
 - activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller, 1240
 - activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller, 1224
 - activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller, 1232
 - activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller, 1236
- ConsumerState
 - activemq::state::ConsumerState, 1243
- contains
 - decaf::util::AbstractCollection, 124
 - decaf::util::Collection, 986
 - decaf::util::concurrent::SynchronousQueue, 2949
 - decaf::util::StlList, 2840
- containsAll
 - decaf::util::AbstractCollection, 124
 - decaf::util::Collection, 986
- decaf::util::concurrent::SynchronousQueue, 2949

- containsKey
 - decaf::util::concurrent::ConcurrentStlMap, 1029
 - decaf::util::Map, 1972
 - decaf::util::StlMap, 2850
- containsValue
 - decaf::util::concurrent::ConcurrentStlMap, 1030
 - decaf::util::Map, 1973
 - decaf::util::StlMap, 2850
- content
 - activemq::commands::Message, 2035
- ControlCommand
 - activemq::commands::ControlCommand, 1244
- ControlCommandMarshaller
 - activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller, 1255
 - activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller, 1263
 - activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller, 1247
 - activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller, 1251
 - activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller, 1259
- convert
 - activemq::util::PrimitiveValueConverter, 2397
 - decaf::util::concurrent::TimeUnit, 3008
- convertConsumerId
 - activemq::wireformat::stomp::StompHelper, 2880
- convertDestination
 - activemq::wireformat::stomp::StompHelper, 2880
- convertMessageId
 - activemq::wireformat::stomp::StompHelper, 2881
- convertProducerId
 - activemq::wireformat::stomp::StompHelper, 2881
- convertProperties
 - activemq::wireformat::stomp::StompHelper, 2882
- convertToCMSException
 - activemq::exceptions::ActiveMQException, 271
- convertTransactionId
 - activemq::wireformat::stomp::StompHelper, 2882
- copy
 - activemq::commands::ActiveMQQueue, 380
 - activemq::commands::ActiveMQTempQueue, 479
 - activemq::commands::ActiveMQTempTopic, 502
 - activemq::commands::ActiveMQTopic, 551
 - activemq::util::ActiveMQProperties, 376
 - activemq::wireformat::stomp::StompFrame, 2875
 - cms::CMSProperties, 966
 - cms::Destination, 1389
 - decaf::util::AbstractCollection, 125
 - decaf::util::concurrent::ConcurrentStlMap, 1030
 - decaf::util::Map, 1974
 - decaf::util::Properties, 2497
 - decaf::util::StlList, 2840
 - decaf::util::StlMap, 2850
 - decaf::util::StlSet, 2870
 - CopyDataStructureMarshaller, 1243
 - activemq::commands::ActiveMQBlobMessage, 343
 - activemq::commands::ActiveMQBytesMessage, 370
 - activemq::commands::ActiveMQDestination, 242
 - activemq::commands::ActiveMQMapMessage, 276
 - activemq::commands::ActiveMQMessage, 306
 - activemq::commands::ActiveMQObjectMessage, 346
 - activemq::commands::ActiveMQQueue, 381
 - activemq::commands::ActiveMQStreamMessage, 425
 - activemq::commands::ActiveMQTempDestination, 457
 - activemq::commands::ActiveMQTempQueue, 479
 - activemq::commands::ActiveMQTempTopic, 503
 - activemq::commands::ActiveMQTextMessage, 527
 - activemq::commands::ActiveMQTopic, 551
 - activemq::commands::BaseCommand, 598
 - activemq::commands::BaseDataStructure, 661
 - activemq::commands::BooleanExpression, 679
 - activemq::commands::BrokerError, 687
 - activemq::commands::BrokerId, 692
 - activemq::commands::BrokerInfo, 715
 - activemq::commands::ConnectionControl, 1057

- activemq::commands::ConnectionError, 1081
- activemq::commands::ConnectionId, 1107
- activemq::commands::ConnectionInfo, 1130
- activemq::commands::ConsumerControl, 1167
- activemq::commands::ConsumerId, 1192
- activemq::commands::ConsumerInfo, 1217
- activemq::commands::ControlCommand, 1244
- activemq::commands::DataArrayResponse, 1269
- activemq::commands::DataResponse, 1308
- activemq::commands::DataStructure, 1373
- activemq::commands::DestinationInfo, 1392
- activemq::commands::DiscoveryEvent, 1418
- activemq::commands::ExceptionResponse, 1485
- activemq::commands::FlushCommand, 1574
- activemq::commands::IntegerResponse, 1668
- activemq::commands::JournalQueueAck, 1719
- activemq::commands::JournalTopicAck, 1743
- activemq::commands::JournalTrace, 1768
- activemq::commands::JournalTransaction, 1791
- activemq::commands::KeepAliveInfo, 1814
- activemq::commands::LastPartialCommand, 1841
- activemq::commands::LocalTransactionId, 1876
- activemq::commands::Message, 2023
- activemq::commands::MessageAck, 2056
- activemq::commands::MessageDispatch, 2085
- activemq::commands::MessageDispatchNotification, 2117
- activemq::commands::MessageId, 2144
- activemq::commands::MessagePull, 2204
- activemq::commands::NetworkBridgeFilter, 2248
- activemq::commands::PartialCommand, 2320
- activemq::commands::ProducerAck, 2421
- activemq::commands::ProducerId, 2448
- activemq::commands::ProducerInfo, 2471
- activemq::commands::RemoveInfo, 2538
- activemq::commands::RemoveSubscriptionInfo, 2561
- activemq::commands::ReplayCommand, 2585
- activemq::commands::Response, 2612
- activemq::commands::SessionId, 2679
- activemq::commands::SessionInfo, 2703
- activemq::commands::ShutdownInfo, 2758
- activemq::commands::SubscriptionInfo, 2909
- activemq::commands::TransactionId, 3017
- activemq::commands::TransactionInfo, 3038
- activemq::commands::WireFormatInfo, 3156
- activemq::commands::XATransactionId, 3194
- correlationId
 - activemq::commands::Message, 2035
 - activemq::commands::MessagePull, 2207
 - activemq::commands::Response, 2614
- countDown
 - decaf::util::concurrent::CountDownLatch, 1267
- CountDownLatch
 - decaf::util::concurrent::CountDownLatch, 1266
- CounterType
 - decaf::lang::Pointer, 2345
- countTokens
 - decaf::util::StringTokenizer, 2905
- create
 - activemq::transport::failover::FailoverTransportFactory, 1524
 - activemq::transport::mock::MockTransportFactory, 2238
 - activemq::transport::tcp::TcpTransportFactory, 2970
 - activemq::transport::TransportFactory, 3072
 - activemq::util::CMSExceptionSupport, 964
 - decaf::internal::util::concurrent::ConditionImpl, 1048
 - decaf::internal::util::concurrent::MutexImpl, 2245
 - decaf::net::URI, 3101
- createBrowser
 - activemq::cmsutil::PooledSession, 2354, 2355
 - activemq::core::ActiveMQSession, 407
 - cms::Session, 2668
- createByteBuffer
 - decaf::internal::nio::BufferFactory, 764
- createBytesMessage

- activemq::cmsutil::PooledSession, 2355
- activemq::core::ActiveMQSession, 408
- cms::Session, 2668, 2669
- createCachedConsumer
 - activemq::cmsutil::PooledSession, 2356
- createCachedProducer
 - activemq::cmsutil::PooledSession, 2356
- createCharBuffer
 - decaf::internal::nio::BufferFactory, 765
- createCMSConnectionFactory
 - cms::ConnectionFactory, 1104
- createComposite
 - activemq::transport::failover::FailoverTransportFactory, 1524
 - activemq::transport::mock::MockTransportFactory, 2238
 - activemq::transport::tcp::TcpTransportFactory, 2970
 - activemq::transport::TransportFactory, 3072
- createConnection
 - activemq::cmsutil::CmsAccessor, 955
 - activemq::core::ActiveMQConnectionFactory, 213, 214
 - cms::ConnectionFactory, 1104, 1105
- createConsumer
 - activemq::cmsutil::PooledSession, 2356, 2357
 - activemq::core::ActiveMQSession, 408, 409
 - cms::Session, 2669, 2670
- createDestination
 - activemq::commands::ActiveMQDestination, 243
- createDoubleBuffer
 - decaf::internal::nio::BufferFactory, 766
- createDurableConsumer
 - activemq::cmsutil::PooledSession, 2358
 - activemq::core::ActiveMQSession, 409
 - cms::Session, 2670
- createFloatBuffer
 - decaf::internal::nio::BufferFactory, 767
- createIntBuffer
 - decaf::internal::nio::BufferFactory, 768
- createLongBuffer
 - decaf::internal::nio::BufferFactory, 769
- createMapMessage
 - activemq::cmsutil::PooledSession, 2358
 - activemq::core::ActiveMQSession, 410
 - cms::Session, 2671
- createMessage
 - activemq::cmsutil::MessageCreator, 2083
 - activemq::cmsutil::PooledSession, 2359
 - activemq::core::ActiveMQSession, 410
 - cms::Session, 2671
- createMessageEOFException
 - activemq::util::CMSExceptionSupport, 964
- createMessageFormatException
 - activemq::util::CMSExceptionSupport, 964
- createNegotiator
 - activemq::wireformat::openwire::OpenWireFormat, 2300
 - activemq::wireformat::stomp::StompWireFormat, 2884
 - activemq::wireformat::WireFormat, 3149
- createObject
 - activemq::wireformat::openwire::marshal::DataStreamMarshal, 1331
 - activemq::wireformat::openwire::marshal::v1::ActiveMQBlobM, 152
 - activemq::wireformat::openwire::marshal::v1::ActiveMQBytesI, 187
 - activemq::wireformat::openwire::marshal::v1::ActiveMQMapM, 290
 - activemq::wireformat::openwire::marshal::v1::ActiveMQMessa, 312
 - activemq::wireformat::openwire::marshal::v1::ActiveMQObject, 352
 - activemq::wireformat::openwire::marshal::v1::ActiveMQQueue, 388
 - activemq::wireformat::openwire::marshal::v1::ActiveMQStream, 441
 - activemq::wireformat::openwire::marshal::v1::ActiveMQTemp, 486
 - activemq::wireformat::openwire::marshal::v1::ActiveMQTemp, 510
 - activemq::wireformat::openwire::marshal::v1::ActiveMQTextM, 535
 - activemq::wireformat::openwire::marshal::v1::ActiveMQTopic, 558
 - activemq::wireformat::openwire::marshal::v1::BrokerIdMarsha, 699
 - activemq::wireformat::openwire::marshal::v1::BrokerInfoMarsh, 727
 - activemq::wireformat::openwire::marshal::v1::ConnectionCont, 1065
 - activemq::wireformat::openwire::marshal::v1::ConnectionError, 1088
 - activemq::wireformat::openwire::marshal::v1::ConnectionIdMa, 1114
 - activemq::wireformat::openwire::marshal::v1::ConnectionInfoM, 1144
 - activemq::wireformat::openwire::marshal::v1::ConsumerContro, 1180
 - activemq::wireformat::openwire::marshal::v1::ConsumerIdMar, 1204
 - activemq::wireformat::openwire::marshal::v1::ConsumerInfoM, 1228

activemq::wireformat::openwire::marshal::v1::CommandMarshaler	2604
activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaler	1255
activemq::wireformat::openwire::marshal::v1::DataResponseMarshaler	2634
activemq::wireformat::openwire::marshal::v1::ResponseMarshaler	1280
activemq::wireformat::openwire::marshal::v1::SessionIdMarshaler	2687
activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaler	1315
activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaler	1408
activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaler	1437
activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaler	1492
activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaler	1581
activemq::wireformat::openwire::marshal::v1::XATransactionInfoMarshaler	1675
activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMarshaler	3050
activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMarshaler	3172
activemq::wireformat::openwire::marshal::v1::ActiveMQMapMarshaler	1735
activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaler	3210
activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMarshaler	1752
activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaler	164
activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMarshaler	1783
activemq::wireformat::openwire::marshal::v1::ActiveMQTemp5Marshaler	199
activemq::wireformat::openwire::marshal::v1::ActiveMQTemp7Marshaler	302
activemq::wireformat::openwire::marshal::v1::ActiveMQTextMarshaler	1806
activemq::wireformat::openwire::marshal::v1::ActiveMQTopic1Marshaler	324
activemq::wireformat::openwire::marshal::v1::ActiveMQTopic5Marshaler	1833
activemq::wireformat::openwire::marshal::v1::ActiveMQTopic7Marshaler	364
activemq::wireformat::openwire::marshal::v1::ActiveMQTopic10Marshaler	1848
activemq::wireformat::openwire::marshal::v1::ActiveMQTopic15Marshaler	400
activemq::wireformat::openwire::marshal::v1::ActiveMQTopic20Marshaler	2077
activemq::wireformat::openwire::marshal::v1::ActiveMQTopic25Marshaler	453
activemq::wireformat::openwire::marshal::v1::ActiveMQTopic30Marshaler	2109
activemq::wireformat::openwire::marshal::v1::ActiveMQTopic35Marshaler	498
activemq::wireformat::openwire::marshal::v1::ActiveMQTopic40Marshaler	522
activemq::wireformat::openwire::marshal::v1::ActiveMQTopic45Marshaler	2133
activemq::wireformat::openwire::marshal::v1::ActiveMQTopic50Marshaler	547
activemq::wireformat::openwire::marshal::v1::ActiveMQTopic55Marshaler	2163
activemq::wireformat::openwire::marshal::v1::ActiveMQTopic60Marshaler	570
activemq::wireformat::openwire::marshal::v1::ActiveMQTopic65Marshaler	2216
activemq::wireformat::openwire::marshal::v1::ActiveMQTopic70Marshaler	711
activemq::wireformat::openwire::marshal::v1::ActiveMQTopic75Marshaler	2263
activemq::wireformat::openwire::marshal::v1::ActiveMQTopic80Marshaler	739
activemq::wireformat::openwire::marshal::v1::ActiveMQTopic85Marshaler	2336
activemq::wireformat::openwire::marshal::v1::ActiveMQTopic90Marshaler	1077
activemq::wireformat::openwire::marshal::v1::ActiveMQTopic95Marshaler	2441
activemq::wireformat::openwire::marshal::v1::ActiveMQTopic100Marshaler	1100
activemq::wireformat::openwire::marshal::v1::ActiveMQTopic105Marshaler	2463
activemq::wireformat::openwire::marshal::v1::ActiveMQTopic110Marshaler	1126
activemq::wireformat::openwire::marshal::v1::ActiveMQTopic115Marshaler	2491
activemq::wireformat::openwire::marshal::v1::ActiveMQTopic120Marshaler	1152
activemq::wireformat::openwire::marshal::v1::ActiveMQTopic125Marshaler	2541
activemq::wireformat::openwire::marshal::v1::ActiveMQTopic130Marshaler	1188
activemq::wireformat::openwire::marshal::v1::ActiveMQTopic135Marshaler	2573

activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller 2549
 1212
 activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller 2569
 1240
 activemq::wireformat::openwire::marshal::v2::ContainerCommandMarshaller 2588
 1263
 activemq::wireformat::openwire::marshal::v2::DataActiveResponseMarshaller 2621
 1288
 activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller 2691
 1327
 activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller 2706
 1396
 activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller 2773
 1421
 activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller 2928
 1488
 activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller 3058
 1577
 activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller 3164
 1671
 activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller 3198
 1723
 activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller 148
 1748
 activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller 183
 1771
 activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller 286
 1794
 activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller 308
 1817
 activemq::wireformat::openwire::marshal::v2::LastReceivedCommandMarshaller 348
 1844
 activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller 384
 1879
 activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller 437
 2061
 activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller 482
 2097
 activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller 506
 2121
 activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller 531
 2147
 activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller 554
 2208
 activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller 695
 2251
 activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller 723
 2323
 activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller 1061
 2425
 activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller 1084
 2451
 activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller 1110
 2475

activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller	2455
1136	
activemq::wireformat::openwire::marshal::v3::ConsumerGroupWithMarshaller	2479
1172	
activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller	2553
1196	
activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller	2577
1224	
activemq::wireformat::openwire::marshal::v3::ConsumerQueueWithMarshaller	2592
1247	
activemq::wireformat::openwire::marshal::v3::DataActiveResponseMarshaller	2625
1272	
activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller	2699
1311	
activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller	2722
1400	
activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller	2769
1425	
activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller	2916
1496	
activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller	3046
1585	
activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller	3180
1679	
activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller	3202
1731	
activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller	156
1756	
activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller	191
1779	
activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller	294
1798	
activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller	316
1825	
activemq::wireformat::openwire::marshal::v3::LastBatchCommandMarshaller	356
1852	
activemq::wireformat::openwire::marshal::v3::LocalTransactionWithMarshaller	392
1883	
activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller	445
2069	
activemq::wireformat::openwire::marshal::v3::MessageDispatchWithMarshaller	490
2105	
activemq::wireformat::openwire::marshal::v3::MessageDispatchWithNotificationMarshaller	514
2129	
activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller	539
2155	
activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller	562
2212	
activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller	703
2255	
activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller	731
2327	
activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller	1069
2429	

activemq::wireformat::openwire::marshal::v4::ConnectionFromMarshaller; openwire::marshal::v4::PartialCommandMarshaller 1092 2332
 activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller; openwire::marshal::v4::ProducerAckMarshaller 1118 2433
 activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller; openwire::marshal::v4::ProducerIdMarshaller 1140 2467
 activemq::wireformat::openwire::marshal::v4::ConsumerFromMarshaller; openwire::marshal::v4::ProducerInfoMarshaller 1176 2487
 activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller; openwire::marshal::v4::RemoveInfoMarshaller 1200 2557
 activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller; openwire::marshal::v4::RemoveSubscriptionMarshaller 1232 2581
 activemq::wireformat::openwire::marshal::v4::ContractCommandMarshaller; openwire::marshal::v4::ReplayCommandMarshaller 1251 2600
 activemq::wireformat::openwire::marshal::v4::DataActiveResponseFromMarshaller; openwire::marshal::v4::ResponseMarshaller 1276 2639
 activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller; openwire::marshal::v4::SessionIdMarshaller 1323 2695
 activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller; openwire::marshal::v4::SessionInfoMarshaller 1404 2714
 activemq::wireformat::openwire::marshal::v4::DisconnectFromMarshaller; openwire::marshal::v4::ShutdownInfoMarshaller 1429 2765
 activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller; openwire::marshal::v4::SubscriptionInfoMarshaller 1500 2924
 activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller; openwire::marshal::v4::TransactionInfoMarshaller 1589 3054
 activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller; openwire::marshal::v4::WireFormatInfoMarshaller 1683 3176
 activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller; openwire::marshal::v4::XATransactionIdMarshaller 1727 3206
 activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller; openwire::marshal::v5::ActiveMQBlobMarshaller 1760 160
 activemq::wireformat::openwire::marshal::v4::JournalTransactioMarshaller; openwire::marshal::v5::ActiveMQBytesMarshaller 1775 195
 activemq::wireformat::openwire::marshal::v4::JournalTransactioMarshaller; openwire::marshal::v5::ActiveMQMapMarshaller 1802 298
 activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller; openwire::marshal::v5::ActiveMQMessageMarshaller 1829 320
 activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller; openwire::marshal::v5::ActiveMQObjectMarshaller 1856 360
 activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller; openwire::marshal::v5::ActiveMQQueueMarshaller 1887 396
 activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller; openwire::marshal::v5::ActiveMQStreamMarshaller 2073 449
 activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller; openwire::marshal::v5::ActiveMQTempMarshaller 2101 494
 activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller; openwire::marshal::v5::ActiveMQTempMarshaller 2125 518
 activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller; openwire::marshal::v5::ActiveMQTextMarshaller 2151 543
 activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller; openwire::marshal::v5::ActiveMQTopicMarshaller 2224 566
 activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller; openwire::marshal::v5::BrokerIdMarshaller 2267 707

- activemq::core::ActiveMQSession, 411
- cms::Session, 2672
- createTemporaryName
 - activemq::commands::ActiveMQDestination, 243
- createTemporaryQueue
 - activemq::cmsutil::PooledSession, 2360
 - activemq::core::ActiveMQSession, 411
 - cms::Session, 2673
- createTemporaryTopic
 - activemq::cmsutil::PooledSession, 2360
 - activemq::core::ActiveMQSession, 411
 - cms::Session, 2673
- createTextMessage
 - activemq::cmsutil::PooledSession, 2361
 - activemq::core::ActiveMQSession, 411, 412
 - cms::Session, 2673
- createTopic
 - activemq::cmsutil::PooledSession, 2361
 - activemq::core::ActiveMQSession, 412
 - cms::Session, 2674
- createWireFormat
 - activemq::transport::AbstractTransportFactory, 140
 - activemq::wireformat::openwire::OpenWireFormatFactory, 2309
 - activemq::wireformat::stomp::StompWireFormatFactory, 2887
 - activemq::wireformat::WireFormatFactory, 3152
- createX500Principal
 - decaf::security_provider::SecurityProvider, 2647
- criticalSection
 - decaf::util::concurrent::ConditionHandle, 1047
- CUNSUMER_MAXPENDINGMSGLIMIT
 - activemq::core::ActiveMQConstants, 229
- currentTimeMillis
 - decaf::lang::System, 2954
- data
 - activemq::commands::DataArrayResponse, 1270
 - activemq::commands::DataResponse, 1310
 - activemq::commands::PartialCommand, 2322
- DataArrayResponse
 - activemq::commands::DataArrayResponse, 1269
- DataArrayResponseMarshaller
 - activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller, 1280
 - activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller, 1288
 - activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller, 1272
 - activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller, 1276
 - activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller, 1284
- DataStream
 - decaf::io::DataStream, 1293
- DataOutputStream
 - decaf::io::DataOutputStream, 1302
- DataResponse
 - activemq::commands::DataResponse, 1308
- DataResponseMarshaller
 - activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller, 1315
 - activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller, 1327
 - activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller, 1311
 - activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller, 1323
 - activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller, 1319
- dataStructure
 - activemq::commands::Message, 2035
- Date
 - decaf::util::Date, 1378
- DAYS
 - decaf::util::concurrent::TimeUnit, 3013
- Debug
 - decaf::util::logging, 117
- debug
 - decaf::util::logging::Logger, 1910
 - decaf::util::logging::SimpleLogger, 2785
- decaf, 102
- decaf/lang/exceptions/ExceptionDefines.h
 - DECAF_CATCH_EXCEPTION_-CONVERT, 3274
 - DECAF_CATCH_NOTHROW, 3275
 - DECAF_CATCH_RETHROW, 3275
 - DECAF_CATCHALL_NOTHROW, 3275
 - DECAF_CATCHALL_THROW, 3276
- decaf/util/Config.h
 - DECAF_API, 3306
 - DECAF_UNUSED, 3306
 - HAVE_PTHREAD_H, 3306
 - HAVE_UUID_T, 3306
 - HAVE_UUID_UUID_H, 3306
- decaf::internal, 102
- decaf::internal::AprPool, 578
- decaf::internal::AprPool, 579

- cleanup, 579
- getAprPool, 579
- getGlobalPool, 579
- decaf::internal::DecafRuntime, 1381
 - ~DecafRuntime, 1382
 - DecafRuntime, 1382
 - getGlobalPool, 1382
- decaf::internal::io, 103
- decaf::internal::io::StandardErrorOutputStream, 2812
 - ~StandardErrorOutputStream, 2814
 - close, 2814
 - flush, 2814
 - lock, 2814
 - notify, 2814
 - notifyAll, 2815
 - StandardErrorOutputStream, 2814
 - tryLock, 2815
 - unlock, 2815
 - wait, 2815, 2816
 - write, 2817
- decaf::internal::io::StandardInputStream, 2818
 - ~StandardInputStream, 2820
 - available, 2820
 - close, 2820
 - lock, 2820
 - mark, 2820
 - markSupported, 2820
 - notify, 2821
 - notifyAll, 2821
 - read, 2821, 2822
 - reset, 2822
 - skip, 2822
 - StandardInputStream, 2820
 - tryLock, 2823
 - unlock, 2823
 - wait, 2823, 2824
- decaf::internal::io::StandardOutputStream, 2825
 - ~StandardOutputStream, 2826
 - close, 2826
 - flush, 2826
 - lock, 2827
 - notify, 2827
 - notifyAll, 2827
 - StandardOutputStream, 2826
 - tryLock, 2827
 - unlock, 2828
 - wait, 2828, 2829
 - write, 2829, 2830
- decaf::internal::net, 103
- decaf::internal::net::URIEncoderDecoder, 3107
 - ~URIEncoderDecoder, 3108
 - decode, 3108
 - encodeOthers, 3108
 - quoteIllegal, 3109
 - URIEncoderDecoder, 3108
 - validate, 3109
 - validateSimple, 3109
- decaf::internal::net::URIHelper, 3110
 - ~URIHelper, 3112
 - isValidDomainName, 3112
 - isValidHexChar, 3112
 - isValidHost, 3112
 - isValidIP4Word, 3113
 - isValidIP6Address, 3113
 - isValidIPv4Address, 3113
 - parseAuthority, 3114
 - parseURI, 3114
 - URIHelper, 3112
 - validateAuthority, 3114
 - validateFragment, 3115
 - validatePath, 3115
 - validateQuery, 3115
 - validateScheme, 3116
 - validateSsp, 3116
 - validateUserInfo, 3116
- decaf::internal::net::URIType, 3126
 - ~URIType, 3128
 - getAuthority, 3128
 - getFragment, 3128
 - getHost, 3128
 - getPath, 3128
 - getPort, 3129
 - getQuery, 3129
 - getScheme, 3129
 - getSchemeSpecificPart, 3129
 - getSource, 3129
 - getUserInfo, 3129
 - isAbsolute, 3130
 - isOpaque, 3130
 - isServerAuthority, 3130
 - isValid, 3130
 - setAbsolute, 3130
 - setAuthority, 3130
 - setFragment, 3131
 - setHost, 3131
 - setOpaque, 3131
 - setPath, 3131
 - setPort, 3131
 - setQuery, 3131
 - setScheme, 3132
 - setSchemeSpecificPart, 3132
 - setServerAuthority, 3132
 - setSource, 3132
 - setUserInfo, 3132
 - setValid, 3133
 - URIType, 3128

- decaf::internal::nio, 103
- decaf::internal::nio::BufferFactory, 762
 - ~BufferFactory, 764
 - createByteBuffer, 764
 - createCharBuffer, 765
 - createDoubleBuffer, 766
 - createFloatBuffer, 767
 - createIntBuffer, 768
 - createLongBuffer, 769
 - createShortBuffer, 770, 771
- decaf::internal::nio::ByteBuffer, 806
 - ~ByteBuffer, 812
 - array, 812
 - arrayOffset, 812
 - asCharBuffer, 813
 - asDoubleBuffer, 813
 - asFloatBuffer, 813
 - asIntBuffer, 814
 - asLongBuffer, 814
 - asReadOnlyBuffer, 814
 - asShortBuffer, 815
 - ByteBuffer, 811
 - compact, 815
 - duplicate, 816
 - get, 816
 - getChar, 817
 - getDouble, 817
 - getFloat, 818
 - getInt, 818, 819
 - getLong, 819
 - getShort, 820
 - hasArray, 820
 - isReadOnly, 821
 - put, 821
 - putChar, 822
 - putDouble, 823
 - putFloat, 823, 824
 - putInt, 824, 825
 - putLong, 825, 826
 - putShort, 826
 - setReadOnly, 827
 - slice, 827
- decaf::internal::nio::ByteArrayPerspective, 843
 - ~ByteArrayPerspective, 847
 - ByteArrayPerspective, 845–847
 - getReferences, 847
 - returnRef, 847
 - takeRef, 847
- decaf::internal::nio::CharArrayBuffer, 922
 - ~CharArrayBuffer, 925
 - _array, 930
 - array, 925
 - arrayOffset, 925
 - asReadOnlyBuffer, 926
 - CharArrayBuffer, 924
 - compact, 926
 - duplicate, 926
 - get, 927
 - hasArray, 927
 - isReadOnly, 928
 - offset, 930
 - put, 928
 - readOnly, 930
 - setReadOnly, 929
 - slice, 929
 - subSequence, 929
- decaf::internal::nio::DoubleArrayBuffer, 1452
 - ~DoubleArrayBuffer, 1455
 - array, 1455
 - arrayOffset, 1455
 - asReadOnlyBuffer, 1456
 - compact, 1456
 - DoubleArrayBuffer, 1454
 - duplicate, 1456
 - get, 1457
 - hasArray, 1457
 - isReadOnly, 1458
 - put, 1458
 - setReadOnly, 1459
 - slice, 1459
- decaf::internal::nio::FloatArrayBuffer, 1555
 - ~FloatArrayBuffer, 1557
 - array, 1557
 - arrayOffset, 1558
 - asReadOnlyBuffer, 1558
 - compact, 1558
 - duplicate, 1559
 - FloatArrayBuffer, 1556, 1557
 - get, 1559
 - hasArray, 1560
 - isReadOnly, 1560
 - put, 1560, 1561
 - setReadOnly, 1561
 - slice, 1561
- decaf::internal::nio::IntArrayBuffer, 1634
 - ~IntArrayBuffer, 1637
 - array, 1637
 - arrayOffset, 1637
 - asReadOnlyBuffer, 1637
 - compact, 1638
 - duplicate, 1638
 - get, 1639
 - hasArray, 1639
 - IntArrayBuffer, 1635, 1636
 - isReadOnly, 1639
 - put, 1640
 - setReadOnly, 1640
 - slice, 1641

- decaf::internal::nio::LongArrayBuffer, 1948
 - ~LongArrayBuffer, 1950
 - array, 1951
 - arrayOffset, 1951
 - asReadOnlyBuffer, 1951
 - compact, 1952
 - duplicate, 1952
 - get, 1952, 1953
 - hasArray, 1953
 - isReadOnly, 1953
 - LongArrayBuffer, 1949, 1950
 - put, 1954
 - setReadOnly, 1954
 - slice, 1955
- decaf::internal::nio::ShortArrayBuffer, 2738
 - ~ShortArrayBuffer, 2741
 - array, 2741
 - arrayOffset, 2741
 - asReadOnlyBuffer, 2742
 - compact, 2742
 - duplicate, 2742
 - get, 2743
 - hasArray, 2743
 - isReadOnly, 2744
 - put, 2744
 - setReadOnly, 2745
 - ShortArrayBuffer, 2740, 2741
 - slice, 2745
- decaf::internal::util, 104
- decaf::internal::util::ByteArrayAdapter, 784
 - ~ByteArrayAdapter, 791
 - ByteArrayAdapter, 789–791
 - clear, 791
 - get, 791
 - getByteArray, 792
 - getCapacity, 792
 - getChar, 792
 - getCharArray, 793
 - getCharCapacity, 793
 - getDouble, 793
 - getDoubleArray, 793
 - getDoubleAt, 794
 - getDoubleCapacity, 794
 - getFloat, 794
 - getFloatArray, 795
 - getFloatAt, 795
 - getFloatCapacity, 795
 - getInt, 795
 - getIntArray, 796
 - getIntAt, 796
 - getIntCapacity, 796
 - getLong, 797
 - getLongArray, 797
 - getLongAt, 797
 - getLongCapacity, 798
 - getShort, 798
 - getShortArray, 798
 - getShortAt, 798
 - getShortCapacity, 799
 - operator[], 799
 - put, 800
 - putChar, 800
 - putDouble, 800
 - putDoubleAt, 801
 - putFloat, 801
 - putFloatAt, 802
 - putInt, 802
 - putIntAt, 802
 - putLong, 803
 - putLongAt, 803
 - putShort, 804
 - putShortAt, 804
 - read, 804
 - resize, 805
 - write, 805
- decaf::internal::util::concurrent, 104
- decaf::internal::util::concurrent::ConditionImpl, 1047
 - create, 1048
 - destroy, 1048
 - notify, 1048
 - notifyAll, 1049
 - wait, 1049
- decaf::internal::util::concurrent::MutexImpl, 2244
 - create, 2245
 - destroy, 2245
 - lock, 2245
 - trylock, 2245
 - unlock, 2246
- decaf::internal::util::concurrent::SynchronizableImpl, 2941
 - ~SynchronizableImpl, 2943
 - lock, 2943
 - notify, 2943
 - notifyAll, 2943
 - SynchronizableImpl, 2943
 - tryLock, 2943
 - unlock, 2944
 - wait, 2944, 2945
- decaf::internal::util::concurrent::Transferer, 3062
- decaf::internal::util::concurrent::TransferQueue, 3062
 - ~TransferQueue, 3063
 - transfer, 3063
 - TransferQueue, 3063

- decaf::internal::util::concurrent::TransferStack, 3064
 - ~TransferStack, 3065
 - transfer, 3065
 - TransferStack, 3065
- decaf::internal::util::HexStringParser, 1607
 - ~HexStringParser, 1608
 - HexStringParser, 1608
 - parse, 1608
 - parseDouble, 1608
 - parseFloat, 1609
- decaf::internal::util::TimerTaskHeap, 3002
 - ~TimerTaskHeap, 3003
 - adjustMinimum, 3003
 - decaf::util::TimerTask, 3002
 - deleteIfCancelled, 3003
 - find, 3004
 - insert, 3004
 - isEmpty, 3004
 - peek, 3004
 - remove, 3004
 - reset, 3004
 - size, 3005
 - TimerTaskHeap, 3003
- decaf::io, 105
- decaf::io::BlockingByteArrayInputStream, 666
 - ~BlockingByteArrayInputStream, 668
 - available, 668
 - BlockingByteArrayInputStream, 668
 - close, 668
 - lock, 668
 - mark, 669
 - markSupported, 669
 - notify, 669
 - notifyAll, 669
 - read, 670
 - reset, 670
 - setByteArray, 671
 - skip, 671
 - tryLock, 672
 - unlock, 672
 - wait, 672, 673
- decaf::io::BufferedInputStream, 747
 - ~BufferedInputStream, 749
 - available, 749
 - BufferedInputStream, 748, 749
 - close, 749
 - mark, 749
 - markSupported, 750
 - read, 750
 - reset, 751
 - skip, 751
- decaf::io::BufferedOutputStream, 752
 - ~BufferedOutputStream, 753
 - BufferedOutputStream, 753
 - close, 753
 - flush, 753
 - write, 754
- decaf::io::ByteArrayInputStream, 828
 - ~ByteArrayInputStream, 830
 - available, 830
 - ByteArrayInputStream, 830
 - close, 830
 - lock, 830
 - mark, 831
 - markSupported, 831
 - notify, 831
 - notifyAll, 831
 - read, 832
 - reset, 832
 - setBuffer, 833
 - setByteArray, 833
 - skip, 833
 - tryLock, 834
 - unlock, 834
 - wait, 834, 835
- decaf::io::ByteArrayOutputStream, 836
 - ~ByteArrayOutputStream, 838
 - ByteArrayOutputStream, 837
 - close, 838
 - flush, 838
 - lock, 838
 - notify, 838
 - notifyAll, 839
 - reset, 839
 - setBuffer, 839
 - size, 839
 - toByteArray, 840
 - toByteArrayRef, 840
 - toString, 840
 - tryLock, 840
 - unlock, 840
 - wait, 841
 - write, 842, 843
 - writeTo, 843
- decaf::io::Closeable, 950
 - ~Closeable, 951
 - close, 951
- decaf::io::DataInputStream, 1291
 - ~DataInputStream, 1293
 - DataInputStream, 1293
 - read, 1293, 1294
 - readBoolean, 1295
 - readByte, 1295
 - readChar, 1295
 - readDouble, 1295
 - readFloat, 1296
 - readFully, 1296, 1297

- readInt, 1297
- readLong, 1297
- readShort, 1298
- readString, 1298
- readUnsignedByte, 1298
- readUnsignedShort, 1299
- readUTF, 1299
- skip, 1299
- decaf::io::DataOutputStream, 1300
 - ~DataOutputStream, 1302
 - buffer, 1307
 - DataOutputStream, 1302
 - size, 1302
 - write, 1302, 1303
 - writeBoolean, 1303
 - writeByte, 1303
 - writeBytes, 1304
 - writeChar, 1304
 - writeChars, 1304
 - writeDouble, 1305
 - writeFloat, 1305
 - writeInt, 1305
 - writeLong, 1305
 - writeShort, 1306
 - writeUnsignedShort, 1306
 - writeUTF, 1306
 - written, 1307
- decaf::io::EOFException, 1474
 - ~EOFException, 1475
 - clone, 1476
 - EOFException, 1474, 1475
- decaf::io::FilterInputStream, 1528
 - ~FilterInputStream, 1530
 - available, 1530
 - close, 1530
 - closed, 1536
 - FilterInputStream, 1530
 - inputStream, 1536
 - isClosed, 1530
 - lock, 1531
 - mark, 1531
 - markSupported, 1531
 - mutex, 1536
 - notify, 1531
 - notifyAll, 1532
 - own, 1536
 - read, 1532, 1533
 - reset, 1533
 - skip, 1534
 - tryLock, 1534
 - unlock, 1534
 - wait, 1535, 1536
- decaf::io::FilterOutputStream, 1536
 - ~FilterOutputStream, 1538
 - close, 1539
 - closed, 1543
 - FilterOutputStream, 1538
 - flush, 1539
 - isClosed, 1539
 - lock, 1539
 - mutex, 1543
 - notify, 1539
 - notifyAll, 1540
 - outputStream, 1543
 - own, 1543
 - tryLock, 1540
 - unlock, 1540
 - wait, 1541
 - write, 1542, 1543
- decaf::io::InputStream, 1630
 - ~InputStream, 1631
 - available, 1631
 - mark, 1631
 - markSupported, 1631
 - read, 1632
 - reset, 1633
 - skip, 1633
- decaf::io::InterruptedIOException, 1693
 - ~InterruptedIOException, 1695
 - clone, 1695
 - InterruptedIOException, 1694, 1695
- decaf::io::IOException, 1707
 - ~IOException, 1708
 - clone, 1709
 - IOException, 1707, 1708
- decaf::io::OutputStream, 2316
 - ~OutputStream, 2317
 - flush, 2317
 - write, 2317, 2318
- decaf::io::Reader, 2518
 - ~Reader, 2519
 - getInputStream, 2519
 - read, 2519
 - readByte, 2519
 - setInputStream, 2519
- decaf::io::UnsupportedEncodingException, 3090
 - ~UnsupportedEncodingException, 3092
 - clone, 3092
 - UnsupportedEncodingException, 3091, 3092
- decaf::io::UTFDataFormatException, 3139
 - ~UTFDataFormatException, 3141
 - clone, 3141
 - UTFDataFormatException, 3139, 3140
- decaf::io::Writer, 3187
 - ~Writer, 3187
 - getOutputStream, 3187

- setOutputStream, 3187
 - write, 3187
 - writeByte, 3188
- decaf::lang, 106
 - operator==, 107
- decaf::lang::Appendable, 576
 - ~Appendable, 577
 - append, 577
- decaf::lang::AtomicRefCounter, 587
 - AtomicRefCounter, 588
 - release, 588
 - swap, 589
- decaf::lang::Boolean, 674
 - ~Boolean, 675
 - _FALSE, 678
 - _TRUE, 678
 - Boolean, 675
 - booleanValue, 675
 - compareTo, 675
 - equals, 676
 - operator<, 676
 - operator==, 677
 - parseBoolean, 677
 - toString, 677, 678
 - valueOf, 678
- decaf::lang::Byte, 776
 - ~Byte, 778
 - Byte, 778
 - byteValue, 778
 - compareTo, 778, 779
 - decode, 779
 - doubleValue, 779
 - equals, 780
 - floatValue, 780
 - intValue, 780
 - longValue, 780
 - MAX_VALUE, 784
 - MIN_VALUE, 784
 - operator<, 780, 781
 - operator==, 781
 - parseByte, 782
 - shortValue, 782
 - SIZE, 784
 - toString, 783
 - valueOf, 783, 784
- decaf::lang::Character, 914
 - byteValue, 917
 - Character, 916
 - compareTo, 917
 - digit, 917
 - doubleValue, 918
 - equals, 918
 - floatValue, 918
 - intValue, 918
 - isDigit, 918
 - isISOControl, 919
 - isLetter, 919
 - isLetterOrDigit, 919
 - isLowerCase, 919
 - isUpperCase, 919
 - isWhitespace, 919
 - longValue, 919
 - MAX_RADIX, 921
 - MAX_VALUE, 921
 - MIN_RADIX, 921
 - MIN_VALUE, 921
 - operator<, 919, 920
 - operator==, 920
 - shortValue, 921
 - SIZE, 921
 - toString, 921
 - valueOf, 921
- decaf::lang::CharSequence, 946
 - ~CharSequence, 947
 - charAt, 947
 - length, 947
 - subSequence, 947
 - toString, 948
- decaf::lang::Comparable, 1009
 - ~Comparable, 1010
 - compareTo, 1010
 - equals, 1010
 - operator<, 1010
 - operator==, 1011
- decaf::lang::Double, 1441
 - ~Double, 1443
 - byteValue, 1444
 - compare, 1444
 - compareTo, 1444
 - Double, 1443
 - doubleToLongBits, 1445
 - doubleToRawLongBits, 1445
 - doubleValue, 1446
 - equals, 1446
 - floatValue, 1446
 - intValue, 1446
 - isInfinite, 1447
 - isNaN, 1447
 - longBitsToDouble, 1447
 - longValue, 1448
 - MAX_VALUE, 1451
 - MIN_VALUE, 1451
 - NaN, 1451
 - NEGATIVE_INFINITY, 1452
 - operator<, 1448
 - operator==, 1448, 1449
 - parseDouble, 1449
 - POSITIVE_INFINITY, 1452

- shortValue, 1449
- SIZE, 1452
- toHexString, 1449
- toString, 1450
- valueOf, 1451
- decaf::lang::DYNAMIC_CAST_TOKEN, 1471
- decaf::lang::Exception, 1476
 - ~Exception, 1479
 - buildMessage, 1479
 - cause, 1482
 - clone, 1479
 - Exception, 1478
 - getCause, 1480
 - getMessage, 1480
 - getStackTrace, 1480
 - getStackTraceString, 1480
 - initCause, 1481
 - message, 1482
 - operator=, 1481
 - printStackTrace, 1481
 - setMark, 1481
 - setMessage, 1482
 - setStackTrace, 1482
 - stackTrace, 1482
 - what, 1482
- decaf::lang::exceptions, 108
- decaf::lang::exceptions::ClassCastException, 948
 - ~ClassCastException, 950
 - ClassCastException, 949, 950
 - clone, 950
- decaf::lang::exceptions::IllegalArgumentException, 1613
 - ~IllegalArgumentException, 1615
 - clone, 1615
 - IllegalArgumentException, 1614, 1615
- decaf::lang::exceptions::IllegalMonitorStateException, 1616
 - ~IllegalMonitorStateException, 1617
 - clone, 1618
 - IllegalMonitorStateException, 1616, 1617
- decaf::lang::exceptions::IllegalStateException, 1618
 - ~IllegalStateException, 1620
 - clone, 1620
 - IllegalStateException, 1619, 1620
- decaf::lang::exceptions::IllegalThreadStateException, 1622
 - ~IllegalThreadStateException, 1623
 - clone, 1624
 - IllegalThreadStateException, 1622, 1623
- decaf::lang::exceptions::IndexOutOfBoundsException, 1627
 - ~IndexOutOfBoundsException, 1629
- clone, 1629
- IndexOutOfBoundsException, 1628, 1629
- decaf::lang::exceptions::InterruptedException, 1691
 - ~InterruptedException, 1693
 - clone, 1693
 - InterruptedException, 1692
- decaf::lang::exceptions::InvalidStateException, 1704
 - ~InvalidStateException, 1706
 - clone, 1706
 - InvalidStateException, 1705, 1706
- decaf::lang::exceptions::NoSuchElementException, 2274
 - ~NoSuchElementException, 2276
 - clone, 2276
 - NoSuchElementException, 2275, 2276
- decaf::lang::exceptions::NullPointerException, 2279
 - ~NullPointerException, 2281
 - clone, 2281
 - NullPointerException, 2280, 2281
- decaf::lang::exceptions::NumberFormatException, 2284
 - ~NumberFormatException, 2286
 - clone, 2286
 - NumberFormatException, 2285, 2286
- decaf::lang::exceptions::RuntimeException, 2644
 - ~RuntimeException, 2646
 - clone, 2646
 - RuntimeException, 2645, 2646
- decaf::lang::exceptions::UnsupportedOperationException, 3094
 - ~UnsupportedOperationException, 3096
 - clone, 3096
 - UnsupportedOperationException, 3094, 3095
- decaf::lang::Float, 1544
 - ~Float, 1546
 - byteValue, 1546
 - compare, 1547
 - compareTo, 1547
 - doubleValue, 1547
 - equals, 1548
 - Float, 1546
 - floatToIntBits, 1548
 - floatToRawIntBits, 1548
 - floatValue, 1549
 - intBitsToFloat, 1549
 - intValue, 1549
 - isInfinite, 1550
 - isNaN, 1550
 - longValue, 1550

- MAX_VALUE, 1554
- MIN_VALUE, 1554
- NaN, 1554
- NEGATIVE_INFINITY, 1554
- operator<, 1550, 1551
- operator==, 1551
- parseFloat, 1552
- POSITIVE_INFINITY, 1554
- shortValue, 1552
- SIZE, 1554
- toHexString, 1552
- toString, 1553
- valueOf, 1553, 1554
- decaf::lang::Integer, 1652
 - ~Integer, 1655
 - bitCount, 1656
 - byteValue, 1656
 - compareTo, 1656
 - decode, 1657
 - doubleValue, 1657
 - equals, 1657
 - floatValue, 1658
 - highestOneBit, 1658
 - Integer, 1655
 - intValue, 1658
 - longValue, 1658
 - lowestOneBit, 1658
 - MAX_VALUE, 1666
 - MIN_VALUE, 1666
 - numberOfLeadingZeros, 1659
 - numberOfTrailingZeros, 1659
 - operator<, 1659, 1660
 - operator==, 1660
 - parseInt, 1661
 - reverse, 1661
 - reverseBytes, 1662
 - rotateLeft, 1662
 - rotateRight, 1662
 - shortValue, 1663
 - signum, 1663
 - SIZE, 1666
 - toBinaryString, 1663
 - toHexString, 1664
 - toOctalString, 1664
 - toString, 1664, 1665
 - valueOf, 1665, 1666
- decaf::lang::Iterable, 1715
 - ~Iterable, 1716
 - iterator, 1716
- decaf::lang::Long, 1934
 - ~Long, 1937
 - bitCount, 1937
 - byteValue, 1937
 - compareTo, 1937, 1938
 - decode, 1938
 - doubleValue, 1938
 - equals, 1939
 - floatValue, 1939
 - highestOneBit, 1939
 - intValue, 1940
 - Long, 1937
 - longValue, 1940
 - lowestOneBit, 1940
 - MAX_VALUE, 1947
 - MIN_VALUE, 1947
 - numberOfLeadingZeros, 1940
 - numberOfTrailingZeros, 1941
 - operator<, 1941
 - operator==, 1942
 - parseLong, 1942, 1943
 - reverse, 1943
 - reverseBytes, 1943
 - rotateLeft, 1943
 - rotateRight, 1944
 - shortValue, 1944
 - signum, 1944
 - SIZE, 1947
 - toBinaryString, 1945
 - toHexString, 1945
 - toOctalString, 1945
 - toString, 1946
 - valueOf, 1946, 1947
- decaf::lang::Math, 1999
 - ~Math, 2001
 - abs, 2001, 2002
 - ceil, 2002
 - E, 2014
 - floor, 2003
 - Math, 2001
 - max, 2004, 2005
 - min, 2005–2007
 - PI, 2014
 - pow, 2008
 - random, 2008
 - round, 2009
 - signum, 2009, 2010
 - sqrt, 2011
 - toDegrees, 2014
 - toRadians, 2014
- decaf::lang::Number, 2282
 - ~Number, 2283
 - byteValue, 2283
 - doubleValue, 2283
 - floatValue, 2283
 - intValue, 2283
 - longValue, 2283
 - shortValue, 2284
- decaf::lang::Pointer, 2343

- ~Pointer, 2347
- CounterType, 2345
- dynamicCast, 2347
- get, 2347
- operator*, 2347
- operator->, 2348
- operator=, 2348
- operator==, 2349, 2350
- Pointer, 2345, 2346
- PointerType, 2345
- ReferenceType, 2345
- release, 2349
- reset, 2349
- staticCast, 2349
- swap, 2349
- decaf::lang::PointerComparator, 2350
 - compare, 2351
 - operator(), 2351
- decaf::lang::Runnable, 2642
 - ~Runnable, 2642
 - run, 2642
- decaf::lang::Runtime, 2643
 - ~Runtime, 2643
 - getRuntime, 2643
 - initializeRuntime, 2643, 2644
 - shutdownRuntime, 2644
- decaf::lang::Short, 2729
 - ~Short, 2732
 - byteValue, 2732
 - compareTo, 2732
 - decode, 2732
 - doubleValue, 2733
 - equals, 2733
 - floatValue, 2733
 - intValue, 2733
 - longValue, 2734
 - MAX_VALUE, 2738
 - MIN_VALUE, 2738
 - operator<, 2734
 - operator==, 2734, 2735
 - parseShort, 2735
 - reverseBytes, 2736
 - Short, 2731
 - shortValue, 2736
 - SIZE, 2738
 - toString, 2736
 - valueOf, 2737
- decaf::lang::STATIC_CAST_TOKEN, 2831
- decaf::lang::System, 2953
 - ~System, 2954
 - availableProcessors, 2954
 - currentTimeMillis, 2954
 - getenv, 2954
 - nanoTime, 2955
 - setenv, 2955
 - System, 2954
 - unsetenv, 2955
- decaf::lang::ThreadGroup, 2977
 - ~ThreadGroup, 2977
 - ThreadGroup, 2977
- decaf::lang::Throwable, 2983
 - ~Throwable, 2984
 - clone, 2984
 - getCause, 2985
 - getMessage, 2985
 - getStackTrace, 2985
 - getStackTraceString, 2986
 - initCause, 2986
 - printStackTrace, 2986
 - setMark, 2986
 - Throwable, 2984
- decaf::net, 108
- decaf::net::BindException, 663
 - ~BindException, 665
 - BindException, 664, 665
 - clone, 665
- decaf::net::BufferedSocket, 755
 - ~BufferedSocket, 756
 - BufferedSocket, 756
 - close, 757
 - connect, 757
 - getInputStream, 757
 - getKeepAlive, 757
 - getOutputStream, 758
 - getReceiveBufferSize, 758
 - getReuseAddress, 758
 - getSendBufferSize, 758
 - getSoLinger, 759
 - getSoTimeout, 759
 - isConnected, 759
 - setKeepAlive, 760
 - setReceiveBufferSize, 760
 - setReuseAddress, 760
 - setSendBufferSize, 760
 - setSoLinger, 761
 - setSoTimeout, 761
- decaf::net::ConnectException, 1049
 - ~ConnectException, 1051
 - clone, 1051
 - ConnectException, 1050, 1051
- decaf::net::HttpRetryException, 1610
 - ~HttpRetryException, 1612
 - clone, 1613
 - HttpRetryException, 1611, 1612
- decaf::net::MalformedURLException, 1968
 - ~MalformedURLException, 1969
 - clone, 1970
 - MalformedURLException, 1968, 1969

- decaf::net::NoRouteToHostException, 2269
 - ~NoRouteToHostException, 2271
 - clone, 2271
 - NoRouteToHostException, 2270, 2271
- decaf::net::PortUnreachableException, 2368
 - ~PortUnreachableException, 2369
 - clone, 2370
 - PortUnreachableException, 2368, 2369
- decaf::net::ProtocolException, 2504
 - ~ProtocolException, 2506
 - clone, 2506
 - ProtocolException, 2505, 2506
- decaf::net::ServerSocket, 2661
 - ~ServerSocket, 2662
 - accept, 2662
 - bind, 2662, 2663
 - close, 2663
 - isBound, 2663
 - ServerSocket, 2662
 - SocketAddress, 2662
 - SocketHandle, 2662
- decaf::net::Socket, 2786
 - ~Socket, 2787
 - connect, 2787
 - getInputStream, 2788
 - getKeepAlive, 2788
 - getOutputStream, 2788
 - getReceiveBufferSize, 2788
 - getReuseAddress, 2789
 - getSendBufferSize, 2789
 - getSoLinger, 2789
 - getSoTimeout, 2789
 - isConnected, 2790
 - setKeepAlive, 2790
 - setReceiveBufferSize, 2790
 - setReuseAddress, 2791
 - setSendBufferSize, 2791
 - setSoLinger, 2791
 - setSoTimeout, 2791
 - SocketAddress, 2787
 - SocketHandle, 2787
- decaf::net::SocketError, 2792
 - getErrorCode, 2792
 - getErrorString, 2792
- decaf::net::SocketException, 2793
 - ~SocketException, 2794
 - clone, 2794
 - SocketException, 2793, 2794
- decaf::net::SocketFactory, 2795
 - ~SocketFactory, 2796
 - createSocket, 2796
- decaf::net::SocketInputStream, 2796
 - ~SocketInputStream, 2798
 - available, 2798
 - close, 2798
 - lock, 2798
 - mark, 2799
 - markSupported, 2799
 - notify, 2799
 - notifyAll, 2799
 - read, 2800
 - reset, 2800
 - skip, 2801
 - SocketInputStream, 2798
 - tryLock, 2801
 - unlock, 2801
 - wait, 2802, 2803
- decaf::net::SocketOutputStream, 2803
 - ~SocketOutputStream, 2805
 - close, 2805
 - flush, 2805
 - lock, 2805
 - notify, 2805
 - notifyAll, 2806
 - SocketOutputStream, 2805
 - tryLock, 2806
 - unlock, 2806
 - wait, 2807
 - write, 2808
- decaf::net::SocketTimeoutException, 2809
 - ~SocketTimeoutException, 2811
 - clone, 2811
 - SocketTimeoutException, 2810, 2811
- decaf::net::TcpSocket, 2959
 - ~TcpSocket, 2961
 - checkResult, 2961
 - close, 2961
 - connect, 2961, 2962
 - getInputStream, 2962
 - getKeepAlive, 2962
 - getOutputStream, 2963
 - getReceiveBufferSize, 2963
 - getReuseAddress, 2963
 - getSendBufferSize, 2963
 - getSocketHandle, 2964
 - getSoLinger, 2964
 - getSoTimeout, 2964
 - getTcpNoDelay, 2964
 - isConnected, 2964
 - setKeepAlive, 2965
 - setReceiveBufferSize, 2965
 - setReuseAddress, 2965
 - setSendBufferSize, 2965
 - setSoLinger, 2966
 - setSoTimeout, 2966
 - setTcpNoDelay, 2966
 - TcpSocket, 2961
- decaf::net::UnknownHostException, 3085

- ~UnknownHostException, 3087
- clone, 3087
- UnknownHostException, 3085, 3086
- decaf::net::UnknownServiceException, 3087
 - ~UnknownServiceException, 3089
 - clone, 3089
 - UnknownServiceException, 3088, 3089
- decaf::net::URI, 3096
 - ~URI, 3100
 - compareTo, 3100
 - create, 3101
 - equals, 3101
 - getAuthority, 3101
 - getFragment, 3101
 - getHost, 3101
 - getPath, 3101
 - getPort, 3102
 - getQuery, 3102
 - getRawAuthority, 3102
 - getRawFragment, 3102
 - getRawPath, 3102
 - getRawQuery, 3102
 - getRawSchemeSpecificPart, 3103
 - getRawUserInfo, 3103
 - getScheme, 3103
 - getSchemeSpecificPart, 3103
 - getUserInfo, 3103
 - isAbsolute, 3103
 - isOpaque, 3104
 - normalize, 3104
 - operator<, 3104
 - operator==, 3105
 - parseServerAuthority, 3105
 - relativize, 3105
 - resolve, 3106
 - toString, 3107
 - toURL, 3107
 - URI, 3099, 3100
- decaf::net::URISyntaxException, 3122
 - ~URISyntaxException, 3125
 - clone, 3125
 - getIndex, 3125
 - getInput, 3125
 - getReason, 3125
 - URISyntaxException, 3123, 3124
- decaf::net::URL, 3133
 - ~URL, 3135
 - URL, 3135
- decaf::net::URLDecoder, 3135
 - ~URLDecoder, 3135
 - decode, 3135
- decaf::net::URLEncoder, 3136
 - ~URLEncoder, 3136
 - encode, 3136
- decaf::nio, 109
- decaf::nio::Buffer, 741
 - ~Buffer, 744
 - _capacity, 747
 - _limit, 747
 - _mark, 747
 - _markSet, 747
 - _position, 747
 - Buffer, 744
 - capacity, 744
 - clear, 744
 - flip, 744
 - hasRemaining, 745
 - isReadOnly, 745
 - limit, 745
 - mark, 746
 - position, 746
 - remaining, 746
 - reset, 746
 - rewind, 747
- decaf::nio::BufferOverflowException, 771
 - ~BufferOverflowException, 773
 - BufferOverflowException, 772, 773
 - clone, 773
- decaf::nio::BufferUnderflowException, 774
 - ~BufferUnderflowException, 775
 - BufferUnderflowException, 774, 775
 - clone, 776
- decaf::nio::ByteBuffer, 848
 - ~ByteBuffer, 853
 - allocate, 854
 - array, 854
 - arrayOffset, 854
 - asCharBuffer, 855
 - asDoubleBuffer, 855
 - asFloatBuffer, 855
 - asIntBuffer, 856
 - asLongBuffer, 856
 - asReadOnlyBuffer, 856
 - asShortBuffer, 857
 - ByteBuffer, 853
 - compact, 857
 - compareTo, 857
 - duplicate, 858
 - equals, 858
 - get, 858, 859
 - getChar, 860
 - getDouble, 860, 861
 - getFloat, 861
 - getInt, 862
 - getLong, 862, 863
 - getShort, 863
 - hasArray, 864
 - isReadOnly, 864

- operator<, 864
- operator==, 865
- put, 865–867
- putChar, 867, 868
- putDouble, 868
- putFloat, 869
- putInt, 870
- putLong, 871
- putShort, 871, 872
- slice, 872
- toString, 873
- wrap, 873
- decaf::nio::CharBuffer, 930
 - ~CharBuffer, 934
 - allocate, 934
 - append, 934, 935
 - array, 935
 - arrayOffset, 936
 - asReadOnlyBuffer, 936
 - charAt, 936
 - CharBuffer, 933
 - compact, 937
 - compareTo, 937
 - duplicate, 937
 - equals, 938
 - get, 938, 939
 - hasArray, 939
 - length, 940
 - operator<, 940
 - operator==, 940
 - put, 940–943
 - read, 944
 - slice, 944
 - subSequence, 945
 - toString, 945
 - wrap, 945, 946
- decaf::nio::DoubleBuffer, 1459
 - ~DoubleBuffer, 1462
 - allocate, 1462
 - array, 1462
 - arrayOffset, 1463
 - asReadOnlyBuffer, 1463
 - compact, 1463
 - compareTo, 1464
 - DoubleBuffer, 1462
 - duplicate, 1464
 - equals, 1464
 - get, 1465, 1466
 - hasArray, 1466
 - operator<, 1466
 - operator==, 1467
 - put, 1467–1469
 - slice, 1469
 - toString, 1470
 - wrap, 1470
- decaf::nio::FloatBuffer, 1562
 - ~FloatBuffer, 1564
 - allocate, 1564
 - array, 1565
 - arrayOffset, 1565
 - asReadOnlyBuffer, 1565
 - compact, 1566
 - compareTo, 1566
 - duplicate, 1566
 - equals, 1567
 - FloatBuffer, 1564
 - get, 1567, 1568
 - hasArray, 1568
 - operator<, 1569
 - operator==, 1569
 - put, 1569–1571
 - slice, 1571
 - toString, 1572
 - wrap, 1572
- decaf::nio::IntBuffer, 1641
 - ~IntBuffer, 1644
 - allocate, 1644
 - array, 1644
 - arrayOffset, 1644
 - asReadOnlyBuffer, 1645
 - compact, 1645
 - compareTo, 1646
 - duplicate, 1646
 - equals, 1646
 - get, 1646–1648
 - hasArray, 1648
 - IntBuffer, 1644
 - operator<, 1648
 - operator==, 1648
 - put, 1649, 1650
 - slice, 1651
 - toString, 1651
 - wrap, 1651, 1652
- decaf::nio::InvalidMarkException, 1700
 - ~InvalidMarkException, 1702
 - clone, 1703
 - InvalidMarkException, 1701, 1702
- decaf::nio::LongBuffer, 1955
 - ~LongBuffer, 1958
 - allocate, 1958
 - array, 1958
 - arrayOffset, 1958
 - asReadOnlyBuffer, 1959
 - compact, 1959
 - compareTo, 1960
 - duplicate, 1960
 - equals, 1960
 - get, 1960–1962

- hasArray, 1962
- LongBuffer, 1958
- operator<, 1962
- operator==, 1963
- put, 1963–1965
- slice, 1965
- toString, 1965
- wrap, 1966
- decaf::nio::ReadOnlyBufferException, 2520
 - ~ReadOnlyBufferException, 2522
 - clone, 2522
 - ReadOnlyBufferException, 2520, 2521
- decaf::nio::ShortBuffer, 2745
 - ~ShortBuffer, 2748
 - allocate, 2748
 - array, 2748
 - arrayOffset, 2749
 - asReadOnlyBuffer, 2749
 - compact, 2749
 - compareTo, 2750
 - duplicate, 2750
 - equals, 2750
 - get, 2751, 2752
 - hasArray, 2752
 - operator<, 2752
 - operator==, 2753
 - put, 2753–2755
 - ShortBuffer, 2748
 - slice, 2755
 - toString, 2756
 - wrap, 2756
- decaf::security, 110
- decaf::security::auth, 110
- decaf::security::auth::x500, 110
- decaf::security::auth::x500::X500Principal, 3188
 - ~X500Principal, 3189
 - getEncoded, 3189
 - getName, 3189
 - hashCode, 3189
- decaf::security::cert, 110
- decaf::security::cert::Certificate, 901
 - ~Certificate, 902
 - equals, 902
 - getEncoded, 902
 - getPublicKey, 902, 903
 - getType, 903
 - toString, 903
 - verify, 903, 904
- decaf::security::cert::CertificateEncodingException, 904
 - ~CertificateEncodingException, 906
 - CertificateEncodingException, 905
 - clone, 906
- decaf::security::cert::CertificateException, 906
 - ~CertificateException, 908
 - CertificateException, 907
 - clone, 908
- decaf::security::cert::CertificateExpiredException, 908
 - ~CertificateExpiredException, 910
 - CertificateExpiredException, 909
 - clone, 910
- decaf::security::cert::CertificateNotYetValidException, 910
 - ~CertificateNotYetValidException, 912
 - CertificateNotYetValidException, 911
 - clone, 912
- decaf::security::cert::CertificateParsingException, 912
 - ~CertificateParsingException, 914
 - CertificateParsingException, 913
 - clone, 914
- decaf::security::cert::X509Certificate, 3189
 - ~X509Certificate, 3190
 - checkValidity, 3190
 - getBasicConstraints, 3190
 - getIssuerUniqueID, 3190
 - getIssuerX500Principal, 3190
 - getKeyUsage, 3191
 - getNotAfter, 3191
 - getNotBefore, 3191
 - getSigAlgName, 3191
 - getSigAlgOID, 3191
 - getSigAlgParams, 3191
 - getSignature, 3191
 - getSubjectUniqueID, 3191
 - getSubjectX500Principal, 3192
 - getTBSCertificate, 3192
 - getVersion, 3192
- decaf::security::GeneralSecurityException, 1602
 - ~GeneralSecurityException, 1604
 - clone, 1604
 - GeneralSecurityException, 1602, 1603
- decaf::security::InvalidKeyException, 1698
 - ~InvalidKeyException, 1700
 - clone, 1700
 - InvalidKeyException, 1699, 1700
- decaf::security::Key, 1835
 - ~Key, 1837
 - getAlgorithm, 1837
 - getEncoded, 1837
 - getFormat, 1837
- decaf::security::KeyException, 1837
 - ~KeyException, 1839
 - clone, 1839
 - KeyException, 1838, 1839
- decaf::security::NoSuchAlgorithmException, 2272

- ~NoSuchAlgorithmException, 2274
- clone, 2274
- NoSuchAlgorithmException, 2273
- decaf::security::NoSuchProviderException, 2277
 - ~NoSuchProviderException, 2279
 - clone, 2279
 - NoSuchProviderException, 2278
- decaf::security::Principal, 2411
 - ~Principal, 2412
 - equals, 2412
 - getName, 2412
- decaf::security::PublicKey, 2506
 - ~PublicKey, 2507
- decaf::security::SignatureException, 2779
 - ~SignatureException, 2781
 - clone, 2781
 - SignatureException, 2780, 2781
- decaf::security_provider, 111
- decaf::security_provider::SecurityProvider, 2647
 - ~SecurityProvider, 2647
 - createX500Principal, 2647
- decaf::security_provider::SecurityProviderMap, 2648
 - getInstance, 2648
 - getSecurityProviderNames, 2648
 - lookup, 2649
 - registerSecurityProvider, 2649
 - unregisterSecurityProvider, 2649
- decaf::security_provider::SecurityProviderRegistrar, 2649
 - ~SecurityProviderRegistrar, 2650
 - getProvider, 2650
 - SecurityProviderRegistrar, 2650
- decaf::security_provider::unix, 111
- decaf::security_provider::unix::openssl, 111
- decaf::security_provider::unix::openssl::OpenSSLX500Principal, 2287
 - ~OpenSSLX500Principal, 2288
 - equals, 2289
 - getEncoded, 2289
 - getName, 2289
 - getX509Name, 2289
 - OpenSSLX500Principal, 2288
 - toString, 2290
- decaf::security_provider::unix::openssl::OpenSSLX509Certificate, 2290
 - ~OpenSSLX509Certificate, 2291
 - checkValidity, 2291
 - equals, 2292
 - getBasicConstraints, 2292
 - getEncoded, 2292
 - getIssuerUniqueID, 2292
 - getIssuerX500Principal, 2292
 - getKeyUsage, 2293
 - getNotAfter, 2293
 - getNotBefore, 2293
 - getPublicKey, 2293
 - getSigAlgName, 2293
 - getSigAlgOID, 2294
 - getSigAlgParams, 2294
 - getSignature, 2294
 - getSubjectUniqueID, 2294
 - getSubjectX500Principal, 2294
 - getTBSCertificate, 2294
 - getType, 2294
 - getVersion, 2295
 - toString, 2295
 - verify, 2295
- decaf::util, 112
- decaf::util::AbstractCollection, 119
 - ~AbstractCollection, 122
 - add, 122
 - addAll, 122
 - clear, 123
 - contains, 124
 - containsAll, 124
 - copy, 125
 - equals, 125
 - isEmpty, 125
 - lock, 126
 - mutex, 131
 - notify, 126
 - notifyAll, 126
 - operator=, 126
 - remove, 127
 - removeAll, 127
 - retainAll, 128
 - toArray, 129
 - tryLock, 129
 - wait, 130
- decaf::util::AbstractList, 131
 - ~AbstractList, 132
- decaf::util::AbstractMap, 132
 - ~AbstractMap, 133
- decaf::util::AbstractQueue, 133
 - ~AbstractQueue, 135
 - AbstractQueue, 135
 - add, 135
 - addAll, 135
 - clear, 136
 - element, 136
 - remove, 136
- decaf::util::AbstractSequentialList, 137
 - ~AbstractSequentialList, 138
- decaf::util::AbstractSet, 138
 - ~AbstractSet, 139

- removeAll, 139
- decaf::util::Collection, 982
 - ~Collection, 984
 - add, 984
 - addAll, 984
 - clear, 985
 - contains, 986
 - containsAll, 986
 - equals, 987
 - isEmpty, 987
 - remove, 988
 - removeAll, 988
 - retainAll, 989
 - size, 990
 - toArray, 990
- decaf::util::Comparator, 1011
 - ~Comparator, 1012
 - compare, 1012
 - operator(), 1013
- decaf::util::comparators, 114
- decaf::util::comparators::Less, 1862
 - ~Less, 1863
 - compare, 1863
 - Less, 1863
 - operator(), 1864
- decaf::util::concurrent, 114
- decaf::util::concurrent::atomic, 115
- decaf::util::concurrent::atomic::AtomicBoolean, 579
 - ~AtomicBoolean, 580
 - AtomicBoolean, 580
 - compareAndSet, 580
 - get, 581
 - getAndSet, 581
 - set, 581
 - toString, 581
- decaf::util::concurrent::atomic::AtomicInteger, 582
 - ~AtomicInteger, 583
 - addAndGet, 583
 - AtomicInteger, 583
 - compareAndSet, 584
 - decrementAndGet, 584
 - doubleValue, 584
 - floatValue, 584
 - get, 585
 - getAndAdd, 585
 - getAndDecrement, 585
 - getAndIncrement, 585
 - getAndSet, 585
 - incrementAndGet, 586
 - intValue, 586
 - longValue, 586
 - set, 586
 - toString, 586
- decaf::util::concurrent::atomic::AtomicReference, 589
 - ~AtomicReference, 590
 - AtomicReference, 590
 - compareAndSet, 590
 - get, 590
 - getAndSet, 591
 - set, 591
 - toString, 591
- decaf::util::concurrent::BrokenBarrierException, 683
 - ~BrokenBarrierException, 685
 - BrokenBarrierException, 684, 685
 - clone, 685
- decaf::util::concurrent::Callable, 897
 - ~Callable, 898
 - call, 898
- decaf::util::concurrent::CancellationException, 898
 - ~CancellationException, 900
 - CancellationException, 899, 900
 - clone, 900
- decaf::util::concurrent::ConcurrentMap, 1020
 - ~ConcurrentMap, 1021
 - putIfAbsent, 1021
 - remove, 1022
 - replace, 1023
- decaf::util::concurrent::ConcurrentStlMap, 1024
 - ~ConcurrentStlMap, 1029
 - clear, 1029
 - ConcurrentStlMap, 1029
 - containsKey, 1029
 - containsValue, 1030
 - copy, 1030
 - equals, 1031
 - get, 1031, 1032
 - isEmpty, 1032
 - keySet, 1032
 - lock, 1033
 - notify, 1033
 - notifyAll, 1033
 - put, 1033
 - putAll, 1034
 - putIfAbsent, 1034
 - remove, 1035, 1036
 - replace, 1036, 1037
 - size, 1037
 - tryLock, 1037
 - unlock, 1038
 - values, 1038
 - wait, 1038, 1039
- decaf::util::concurrent::ConditionHandle, 1046
 - ~ConditionHandle, 1047

- condition, 1047
- ConditionHandle, 1047
- criticalSection, 1047
- generation, 1047
- mutex, 1047
- numWaiting, 1047
- numWake, 1047
- semaphore, 1047
- decaf::util::concurrent::CountDownLatch, 1266
 - ~CountDownLatch, 1266
 - await, 1267
 - countDown, 1267
 - CountDownLatch, 1266
 - getCount, 1267
- decaf::util::concurrent::Delayed, 1385
 - ~Delayed, 1386
 - getDelay, 1386
- decaf::util::concurrent::ExecutionException, 1507
 - ~ExecutionException, 1508
 - clone, 1509
 - ExecutionException, 1507, 1508
- decaf::util::concurrent::Executor, 1509
 - ~Executor, 1510
 - execute, 1510
- decaf::util::concurrent::ExecutorService, 1511
 - ~ExecutorService, 1512
 - awaitTermination, 1512
- decaf::util::concurrent::Future, 1597
 - ~Future, 1598
 - cancel, 1598
 - get, 1599
 - isCancelled, 1599
 - isDone, 1600
- decaf::util::concurrent::Lock, 1903
 - ~Lock, 1904
 - isLocked, 1904
 - Lock, 1904
 - lock, 1904
 - unlock, 1904
- decaf::util::concurrent::locks, 116
- decaf::util::concurrent::locks::Condition, 1040
 - ~Condition, 1042
 - await, 1042, 1043
 - awaitNanos, 1043
 - awaitUninterruptibly, 1045
 - awaitUntil, 1045
 - signal, 1046
 - signalAll, 1046
- decaf::util::concurrent::locks::Lock, 1898
 - ~Lock, 1900
 - lock, 1900
 - lockInterruptibly, 1900
 - newCondition, 1901
 - tryLock, 1901, 1902
 - unlock, 1903
- decaf::util::concurrent::locks::LockSupport, 1905
 - ~LockSupport, 1906
 - park, 1906
 - parkNanos, 1906
 - parkUntil, 1907
 - unpark, 1907
- decaf::util::concurrent::locks::ReadWriteLock, 2522
 - ~ReadWriteLock, 2524
 - readLock, 2524
 - writeLock, 2524
- decaf::util::concurrent::locks::ReentrantLock, 2526
 - ~ReentrantLock, 2528
 - getHoldCount, 2528
 - isFair, 2528
 - isHeldByCurrentThread, 2529
 - isLocked, 2529
 - lock, 2529
 - lockInterruptibly, 2530
 - newCondition, 2530
 - ReentrantLock, 2528
 - toString, 2531
 - tryLock, 2531, 2532
 - unlock, 2532
- decaf::util::concurrent::Mutex, 2239
 - ~Mutex, 2240
 - lock, 2240
 - Mutex, 2240
 - notify, 2240
 - notifyAll, 2241
 - tryLock, 2241
 - unlock, 2242
 - wait, 2242, 2243
- decaf::util::concurrent::MutexHandle, 2244
 - ~MutexHandle, 2244
 - lock_count, 2244
 - lock_owner, 2244
 - mutex, 2244
 - MutexHandle, 2244
- decaf::util::concurrent::PooledThread, 2364
 - ~PooledThread, 2364
 - getPooledThreadListener, 2365
 - isBusy, 2365
 - PooledThread, 2364
 - run, 2365
 - setPooledThreadListener, 2365
 - stop, 2365
- decaf::util::concurrent::PooledThreadListener, 2366
 - ~PooledThreadListener, 2367

- onTaskCompleted, 2367
- onTaskException, 2367
- onTaskStarted, 2367
- decaf::util::concurrent::RejectedExecutionException, 2533
 - ~RejectedExecutionException, 2535
 - clone, 2535
 - RejectedExecutionException, 2534, 2535
- decaf::util::concurrent::RejectedExecutionHandler, 2536
 - ~RejectedExecutionHandler, 2536
- decaf::util::concurrent::Semaphore, 2651
 - ~Semaphore, 2654
 - acquire, 2654
 - acquireUninterruptibly, 2655
 - availablePermits, 2656
 - drainPermits, 2656
 - isFair, 2656
 - release, 2656, 2657
 - Semaphore, 2653, 2654
 - toString, 2657
 - tryAcquire, 2657–2659
- decaf::util::concurrent::Synchronizable, 2930
 - ~Synchronizable, 2932
 - lock, 2932
 - notify, 2933
 - notifyAll, 2934
 - tryLock, 2935
 - unlock, 2936
 - wait, 2937, 2938, 2940
- decaf::util::concurrent::SynchronousQueue, 2946
 - ~SynchronousQueue, 2949
 - clear, 2949
 - contains, 2949
 - containsAll, 2949
 - drainTo, 2949
 - equals, 2949
 - isEmpty, 2950
 - iterator, 2950
 - offer, 2950
 - peek, 2951
 - poll, 2951
 - put, 2951
 - remainingCapacity, 2952
 - remove, 2952
 - removeAll, 2952
 - retainAll, 2952
 - size, 2952
 - SynchronousQueue, 2949
 - take, 2952
 - toArray, 2953
- decaf::util::concurrent::TaskListener, 2957
 - ~TaskListener, 2957
- onTaskComplete, 2957
- onTaskException, 2957
- decaf::util::concurrent::ThreadFactory, 2976
 - ~ThreadFactory, 2976
 - newThread, 2976
- decaf::util::concurrent::ThreadPool, 2977
 - ~ThreadPool, 2979
 - DEFAULT_MAX_BLOCK_SIZE, 2983
 - DEFAULT_MAX_POOL_SIZE, 2983
 - deQueueTask, 2979
 - getBacklog, 2979
 - getBlockSize, 2980
 - getFreeThreadCount, 2980
 - getInstance, 2980
 - getMaxThreads, 2980
 - getPoolSize, 2980
 - onTaskCompleted, 2981
 - onTaskException, 2981
 - onTaskStarted, 2981
 - queueTask, 2981
 - reserve, 2982
 - setBlockSize, 2982
 - setMaxThreads, 2982
 - Task, 2979
 - ThreadPool, 2979
- decaf::util::concurrent::TimeoutException, 2987
 - ~TimeoutException, 2989
 - clone, 2989
 - TimeoutException, 2987, 2988
- decaf::util::concurrent::TimeUnit, 3005
 - ~TimeUnit, 3007
 - compareTo, 3007
 - convert, 3008
 - DAYS, 3013
 - equals, 3008
 - HOURS, 3013
 - MICROSECONDS, 3013
 - MILLISECONDS, 3013
 - MINUTES, 3013
 - NANOSECONDS, 3013
 - operator<, 3008
 - operator==, 3009
 - SECONDS, 3013
 - sleep, 3009
 - timedJoin, 3009
 - timedWait, 3010
 - TimeUnit, 3007
 - toDays, 3010
 - toHours, 3011
 - toMicros, 3011
 - toMillis, 3011
 - toMinutes, 3012
 - toNanos, 3012
 - toSeconds, 3012

- toString, 3013
- valueOf, 3013
- values, 3014
- decaf::util::Date, 1377
 - ~Date, 1379
 - after, 1379
 - before, 1379
 - compareTo, 1379
 - Date, 1378
 - equals, 1379
 - getTime, 1380
 - operator<, 1380
 - operator=, 1380
 - operator==, 1380
 - setTime, 1380
 - toString, 1381
- decaf::util::Iterator, 1716
 - ~Iterator, 1717
 - hasNext, 1717
 - next, 1717
 - remove, 1717
- decaf::util::List, 1865
 - ~List, 1867
 - add, 1867
 - addAll, 1867
 - get, 1868
 - indexOf, 1868
 - lastIndexOf, 1868
 - listIterator, 1869, 1870
 - remove, 1870
 - set, 1871
- decaf::util::ListIterator, 1871
 - ~ListIterator, 1872
 - add, 1872
 - hasPrevious, 1873
 - nextIndex, 1873
 - previous, 1873
 - previousIndex, 1873
 - set, 1874
- decaf::util::logging, 116
 - Debug, 117
 - Error, 117
 - Fatal, 117
 - Info, 117
 - Level, 117
 - Markblock, 117
 - Null, 117
 - Off, 117
 - Throwing, 117
 - Warn, 117
- decaf::util::logging::Filter, 1527
 - ~Filter, 1527
 - isLoggable, 1527
- decaf::util::logging::Formatter, 1595
 - ~Formatter, 1596
 - format, 1596
 - formatMessage, 1596
 - getHead, 1597
 - getTail, 1597
- decaf::util::logging::Handler, 1604
 - ~Handler, 1605
 - flush, 1605
 - getFilter, 1605
 - getFormatter, 1606
 - getLevel, 1606
 - isLoggable, 1606
 - publish, 1606
 - setFilter, 1607
 - setFormatter, 1607
 - setLevel, 1607
- decaf::util::logging::Logger, 1908
 - ~Logger, 1910
 - addHandler, 1910
 - debug, 1910
 - entry, 1910
 - error, 1911
 - exit, 1911
 - fatal, 1911
 - getAnonymousLogger, 1912
 - getFilter, 1912
 - getLevel, 1912
 - getLogger, 1912
 - getName, 1913
 - getParentHandlers, 1913
 - info, 1913
 - isLoggable, 1913
 - log, 1914, 1915
 - Logger, 1910
 - removeHandler, 1915
 - setFilter, 1915
 - setLevel, 1915
 - setParentHandlers, 1916
 - warn, 1916
- decaf::util::logging::LoggerHierarchy, 1916
 - ~LoggerHierarchy, 1917
 - LoggerHierarchy, 1917
- decaf::util::logging::LogManager, 1923
 - ~LogManager, 1925
 - addPropertyChangeListener, 1925
 - destroy, 1925
 - getInstance, 1926
 - getLogger, 1926
 - getLoggerNames, 1926
 - getProperties, 1926
 - getProperty, 1926
 - LogManager, 1925
 - operator=, 1927
 - removePropertyChangeListener, 1927

- returnInstance, 1927
- setProperty, 1927
- decaf::util::logging::LogRecord, 1928
 - ~LogRecord, 1929
 - getLevel, 1929
 - getLoggerName, 1929
 - getMessage, 1929
 - getSourceFile, 1929
 - getSourceFunction, 1930
 - getSourceLine, 1930
 - getTimestamp, 1930
 - getTreadId, 1930
 - LogRecord, 1929
 - setLevel, 1930
 - setLoggerName, 1930
 - setMessage, 1931
 - setSourceFile, 1931
 - setSourceFunction, 1931
 - setSourceLine, 1931
 - setTimestamp, 1931
 - setTreadId, 1931
- decaf::util::logging::LogWriter, 1932
 - ~LogWriter, 1933
 - destroy, 1933
 - getInstance, 1933
 - log, 1933
 - LogWriter, 1933
 - returnInstance, 1933
- decaf::util::logging::MarkBlockLogger, 1992
 - ~MarkBlockLogger, 1992
 - MarkBlockLogger, 1992
- decaf::util::logging::PropertiesChangeListener, 2503
 - ~PropertiesChangeListener, 2504
 - onPropertyChanged, 2504
- decaf::util::logging::SimpleFormatter, 2782
 - ~SimpleFormatter, 2783
 - format, 2783
 - formatMessage, 2783
 - getHead, 2783
 - getTail, 2783
 - SimpleFormatter, 2783
- decaf::util::logging::SimpleLogger, 2784
 - ~SimpleLogger, 2785
 - debug, 2785
 - error, 2785
 - fatal, 2785
 - info, 2785
 - log, 2785
 - mark, 2785
 - SimpleLogger, 2785
 - warn, 2785
- decaf::util::logging::StreamHandler, 2888
 - ~StreamHandler, 2889
 - close, 2889
 - flush, 2890
 - getFilter, 2890
 - getFormatter, 2890
 - getLevel, 2890
 - getOutputStream, 2890
 - isLoggable, 2891
 - publish, 2891
 - setFilter, 2891
 - setFormatter, 2891
 - setLevel, 2891
 - StreamHandler, 2889
- decaf::util::Map, 1970
 - ~Map, 1972
 - clear, 1972
 - containsKey, 1972
 - containsValue, 1973
 - copy, 1974
 - equals, 1974
 - get, 1975, 1976
 - isEmpty, 1976
 - keySet, 1977
 - Map, 1972
 - put, 1978
 - putAll, 1979
 - remove, 1980
 - size, 1981
 - values, 1981
- decaf::util::Map::Entry, 1473
 - ~Entry, 1473
 - Entry, 1473
 - getKey, 1473
 - getValue, 1473
 - setValue, 1473
- decaf::util::PriorityQueue, 2413
 - ~PriorityQueue, 2416
 - add, 2416
 - clear, 2416
 - comparator, 2417
 - iterator, 2417
 - offer, 2417
 - operator=, 2418
 - peek, 2418
 - poll, 2418
 - PriorityQueue, 2415, 2416
 - PriorityQueueIterator, 2420
 - remove, 2418, 2419
 - size, 2419
- decaf::util::Properties, 2494
 - ~Properties, 2496
 - clear, 2496
 - clone, 2496
 - copy, 2497
 - defaults, 2503

- equals, 2497
- getProperty, 2497
- hasProperty, 2498
- isEmpty, 2498
- load, 2498, 2500
- operator=, 2500
- Properties, 2496
- remove, 2500
- setProperty, 2501
- size, 2501
- store, 2501, 2502
- toArray, 2502
- toString, 2502
- decaf::util::Queue, 2507
 - ~Queue, 2508
 - element, 2508
 - offer, 2508
 - peek, 2509
 - poll, 2509
 - remove, 2510
- decaf::util::Random, 2513
 - next, 2515
 - nextBoolean, 2515
 - nextBytes, 2515
 - nextDouble, 2515
 - nextFloat, 2516
 - nextGaussian, 2516
 - nextInt, 2516
 - nextLong, 2517
 - Random, 2514
 - setSeed, 2517
- decaf::util::Set, 2729
 - ~Set, 2729
- decaf::util::StlList, 2833
 - ~StlList, 2838
 - add, 2838, 2839
 - addAll, 2839
 - clear, 2840
 - contains, 2840
 - copy, 2840
 - equals, 2841
 - get, 2841
 - indexOf, 2841
 - isEmpty, 2841
 - iterator, 2842
 - lastIndexOf, 2842
 - listIterator, 2842, 2843
 - remove, 2843
 - set, 2844
 - size, 2844
 - StlList, 2837, 2838
- decaf::util::StlMap, 2845
 - ~StlMap, 2849
 - clear, 2850
 - containsKey, 2850
 - containsValue, 2850
 - copy, 2850, 2851
 - equals, 2851
 - get, 2851, 2852
 - isEmpty, 2852
 - keySet, 2852
 - lock, 2853
 - notify, 2853
 - notifyAll, 2853
 - put, 2854
 - putAll, 2854
 - remove, 2854
 - size, 2855
 - StlMap, 2849
 - tryLock, 2855
 - unlock, 2855
 - values, 2856
 - wait, 2856, 2857
- decaf::util::StlQueue, 2858
 - ~StlQueue, 2860
 - back, 2860
 - clear, 2860
 - empty, 2860
 - enqueueFront, 2861
 - front, 2861
 - getSafeValue, 2861
 - iterator, 2861
 - lock, 2862
 - notify, 2862
 - notifyAll, 2862
 - pop, 2862
 - push, 2863
 - reverse, 2863
 - size, 2863
 - StlQueue, 2860
 - toArray, 2863
 - tryLock, 2863
 - unlock, 2864
 - wait, 2864, 2865
- decaf::util::StlSet, 2865
 - ~StlSet, 2868
 - add, 2868
 - clear, 2869
 - contains, 2869
 - copy, 2870
 - equals, 2870
 - isEmpty, 2870
 - iterator, 2870
 - remove, 2870
 - size, 2871
 - StlSet, 2868
- decaf::util::StringTokenizer, 2904
 - ~StringTokenizer, 2905

- countTokens, 2905
- hasMoreTokens, 2905
- nextToken, 2905, 2906
- reset, 2906
- StringTokenizer, 2905
- toArray, 2906
- decaf::util::Timer, 2989
 - ~Timer, 2992
 - cancel, 2992
 - purge, 2992
 - schedule, 2992–2997
 - scheduleAtFixedRate, 2997–2999
 - Timer, 2992
- decaf::util::TimerTask, 3000
 - ~TimerTask, 3001
 - cancel, 3001
 - decaf::internal::util::TimerTaskHeap, 3002
 - getWhen, 3002
 - isScheduled, 3002
 - scheduledExecutionTime, 3002
 - setScheduledTime, 3002
 - Timer, 3002
 - TimerImpl, 3002
 - TimerTask, 3001
- decaf::util::UUID, 3141
 - ~UUID, 3143
 - clockSequence, 3143
 - compareTo, 3144
 - equals, 3144
 - fromString, 3144
 - getLeastSignificantBits, 3144
 - getMostSignificantBits, 3144
 - nameUUIDFromBytes, 3145
 - node, 3145
 - operator<, 3145
 - operator==, 3146
 - randomUUID, 3146
 - timestamp, 3146
 - toString, 3147
 - UUID, 3143
 - variant, 3147
 - version, 3147
- DECAF_API
 - decaf/util/Config.h, 3306
- DECAF_CATCH_EXCEPTION_CONVERT
 - decaf/lang/exceptions/ExceptionDe-
fines.h, 3274
- DECAF_CATCH_NOTHROW
 - decaf/lang/exceptions/ExceptionDe-
fines.h, 3275
- DECAF_CATCH_RETHROW
 - decaf/lang/exceptions/ExceptionDe-
fines.h, 3275
- DECAF_CATCHALL_NOTHROW
 - decaf/lang/exceptions/ExceptionDe-
fines.h, 3275
- DECAF_CATCHALL_THROW
 - decaf/lang/exceptions/ExceptionDe-
fines.h, 3276
- DECAF_UNUSED
 - decaf/util/Config.h, 3306
- DecafRuntime
 - decaf::internal::DecafRuntime, 1382
- decode
 - decaf::internal::net::URLEncoderDecoder,
3108
 - decaf::lang::Byte, 779
 - decaf::lang::Integer, 1657
 - decaf::lang::Long, 1938
 - decaf::lang::Short, 2732
 - decaf::net::URLDecoder, 3135
- decreaseUsage
 - activemq::util::MemoryUsage, 2016
 - activemq::util::Usage, 3137
- decrementAndGet
 - decaf::util::concurrent::atomic::AtomicInteger,
584
- DedicatedTaskRunner
 - activemq::threads::DedicatedTaskRunner,
1383
- DEFAULT_MAX_BLOCK_SIZE
 - decaf::util::concurrent::ThreadPool, 2983
- DEFAULT_MAX_POOL_SIZE
 - decaf::util::concurrent::ThreadPool, 2983
- DEFAULT_MESSAGE_SIZE
 - activemq::commands::Message, 2035
- DEFAULT_ORDERED_TARGET
 - activemq::commands::ActiveMQDestination,
249
- DEFAULT_PRIORITY
 - activemq::cmsutil::CmsTemplate, 981
- DEFAULT_TIME_TO_LIVE
 - activemq::cmsutil::CmsTemplate, 981
- DEFAULT_VERSION
 - activemq::wireformat::openwire::OpenWireFormat,
2308
- defaults
 - decaf::util::Properties, 2503
- deleteIfCancelled
 - decaf::internal::util::TimerTaskHeap, 3003
- deliverAcks
 - activemq::core::ActiveMQConsumer, 234
 - activemq::core::ActiveMQSession, 412
- DELIVERY_MODE
 - cms::DeliveryMode, 1387
- deliverySequenceId
 - activemq::commands::MessageDispatchNotification,
2120

- dequeue
 - activemq::core::ActiveMQConsumer, 234
 - activemq::core::MessageDispatchChannel, 2090
- dequeueNoWait
 - activemq::core::MessageDispatchChannel, 2091
- deQueueTask
 - decaf::util::concurrent::ThreadPool, 2979
- destination
 - activemq::cmsutil::CmsTemplate::ProducerExecutor, 959
 - 2446
 - activemq::cmsutil::CmsTemplate::ReceiveExecutor, 2526
 - activemq::commands::ConsumerInfo, 1222
 - activemq::commands::DestinationInfo, 1395
 - activemq::commands::JournalQueueAck, 1721
 - activemq::commands::JournalTopicAck, 1746
 - activemq::commands::Message, 2035
 - activemq::commands::MessageAck, 2060
 - activemq::commands::MessageDispatch, 2088
 - activemq::commands::MessageDispatchNotification, 2120
 - activemq::commands::MessagePull, 2207
 - activemq::commands::ProducerInfo, 2474
 - activemq::commands::SubscriptionInfo, 2911
- DESTINATION_ADD_OPERATION
 - activemq::core::ActiveMQConstants, 228
- DESTINATION_REMOVE_OPERATION
 - activemq::core::ActiveMQConstants, 228
- DestinationActions
 - activemq::core::ActiveMQConstants, 228
- DestinationInfo
 - activemq::commands::DestinationInfo, 1392
- DestinationInfoMarshaller
 - activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller, 1408
 - activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller, 1396
 - activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller, 1400
 - activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller, 1404
 - activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller, 1412
- DestinationOption
 - activemq::core::ActiveMQConstants, 228
- DestinationType
 - cms::Destination, 1388
- destOptionMap
 - activemq::core::ActiveMQConstants::StaticInitializer, 2832
- destOptions
 - activemq::core::ActiveMQConstants::StaticInitializer, 2832
- destroy
 - activemq::cmsutil::CmsAccessor, 956
 - activemq::cmsutil::CmsDestinationAccessor, 959
 - activemq::cmsutil::CmsTemplate, 973
 - activemq::cmsutil::DestinationResolver, 1415
 - activemq::cmsutil::DynamicDestinationResolver, 1472
 - activemq::cmsutil::ResourceLifecycleManager, 2610
 - activemq::commands::ActiveMQTempQueue, 479
 - activemq::commands::ActiveMQTempTopic, 503
 - cms::TemporaryQueue, 2972
 - cms::TemporaryTopic, 2973
 - decaf::internal::util::concurrent::ConditionImpl, 1048
 - decaf::internal::util::concurrent::MutexImpl, 2245
 - decaf::util::logging::LogManager, 1925
 - decaf::util::logging::LogWriter, 1933
- destroyDestination
 - activemq::core::ActiveMQConnection, 206
- destroyMarshalers
 - activemq::wireformat::openwire::OpenWireFormat, 2300
- digit
 - decaf::lang::Character, 917
- DISCONNECT
 - activemq::wireformat::stomp::StompCommandConstants, 2873
- DiscoveryEvent
 - activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller, 1418
 - activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller, 1417
 - activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller, 1417
 - activemq::wireformat::openwire::marshal::v4::DiscoveryEventMarshaller, 1417
 - activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller, 1425

- dispatch
 - activemq::core::ActiveMQConsumer, 234
 - activemq::core::ActiveMQSession, 412
 - activemq::core::Dispatcher, 1441
- dispatchAsync
 - activemq::commands::ConsumerInfo, 1222
 - activemq::commands::ProducerInfo, 2474
- DispatchData
 - activemq::core::DispatchData, 1440
- disposeOf
 - activemq::core::ActiveMQSession, 413
- doClose
 - activemq::core::ActiveMQConsumer, 234
- doCreateComposite
 - activemq::transport::failover::FailoverTransportFactory, 1524
 - activemq::transport::mock::MockTransportFactory, 2238
 - activemq::transport::tcp::TcpTransportFactory, 2971
- doInCms
 - activemq::cmsutil::CmsTemplate::ProducerExecutor, 2445
 - activemq::cmsutil::CmsTemplate::ReceiveExecutor, 2525
 - activemq::cmsutil::CmsTemplate::SendExecutor, 2661
 - activemq::cmsutil::ProducerCallback, 2444
 - activemq::cmsutil::SessionCallback, 2677
- doStartTransaction
 - activemq::core::ActiveMQSession, 413
- Double
 - decaf::lang::Double, 1443
- DOUBLE_TYPE
 - activemq::util::PrimitiveValueNode, 2402
- DoubleArrayBuffer
 - decaf::internal::nio::DoubleArrayBuffer, 1454
- DoubleBuffer
 - decaf::nio::DoubleBuffer, 1462
- doubleToLongBits
 - decaf::lang::Double, 1445
- doubleToRawLongBits
 - decaf::lang::Double, 1445
- doubleValue
 - activemq::util::PrimitiveValueNode::PrimitiveValueNode, 2396
 - decaf::lang::Byte, 779
 - decaf::lang::Character, 918
 - decaf::lang::Double, 1446
 - decaf::lang::Float, 1547
 - decaf::lang::Integer, 1657
 - decaf::lang::Long, 1938
 - decaf::lang::Number, 2283
 - decaf::lang::Short, 2733
 - decaf::util::concurrent::atomic::AtomicInteger, 584
- doUnmarshal
 - activemq::wireformat::openwire::OpenWireFormat, 2300
- drainPermits
 - decaf::util::concurrent::Semaphore, 2656
- drainTo
 - decaf::util::concurrent::SynchronousQueue, 2949
- droppable
 - activemq::commands::Message, 2035
- duplexConnection
 - activemq::commands::BrokerInfo, 721
- duplicate
 - decaf::internal::nio::ByteBuffer, 816
 - decaf::internal::nio::CharArrayBuffer, 926
 - decaf::internal::nio::DoubleArrayBuffer, 1456
 - decaf::internal::nio::FloatArrayBuffer, 1559
 - decaf::internal::nio::IntArrayBuffer, 1638
 - decaf::internal::nio::LongArrayBuffer, 1952
 - decaf::internal::nio::ShortArrayBuffer, 2742
 - decaf::nio::ByteBuffer, 858
 - decaf::nio::CharBuffer, 937
 - decaf::nio::DoubleBuffer, 1464
 - decaf::nio::FloatBuffer, 1566
 - decaf::nio::IntBuffer, 1646
 - decaf::nio::LongBuffer, 1960
 - decaf::nio::ShortBuffer, 2750
- DUPS_OK_ACKNOWLEDGE
 - cms::Session, 2667
- dynamicCast
 - decaf::lang::Pointer, 2347
- E
 - decaf::lang::Math, 2014
- element
 - decaf::util::AbstractQueue, 136
 - decaf::util::Queue, 2508
- empty
 - decaf::util::StlQueue, 2860
- encode
 - decaf::net::URLEncoder, 3136
- encodeOthers
 - decaf::internal::net::URLEncoderDecoder, 3108
- enqueue
 - activemq::core::MessageDispatchChannel, 2091
- enqueueFirst

- activemq::core::MessageDispatchChannel, 2091
- enqueueFront
 - decaf::util::StlQueue, 2861
- enqueueUsage
 - activemq::util::MemoryUsage, 2016
 - activemq::util::Usage, 3137
- Entry
 - decaf::util::Map::Entry, 1473
- entry
 - decaf::util::logging::Logger, 1910
- EOFException
 - decaf::io::EOFException, 1474, 1475
- equals
 - activemq::commands::ActiveMQBlobMessage, 144
 - activemq::commands::ActiveMQBytesMessage, 171
 - activemq::commands::ActiveMQDestination, 243
 - activemq::commands::ActiveMQMapMessage, 276
 - activemq::commands::ActiveMQMessage, 306
 - activemq::commands::ActiveMQMessageTemplate, 331
 - activemq::commands::ActiveMQObjectMessage, 346
 - activemq::commands::ActiveMQQueue, 381
 - activemq::commands::ActiveMQStreamMessage, 426
 - activemq::commands::ActiveMQTempDestination, 457
 - activemq::commands::ActiveMQTempQueue, 479
 - activemq::commands::ActiveMQTempTopic, 503
 - activemq::commands::ActiveMQTextMessage, 527
 - activemq::commands::ActiveMQTopic, 551
 - activemq::commands::BaseCommand, 598
 - activemq::commands::BaseDataStructure, 661
 - activemq::commands::BooleanExpression, 680
 - activemq::commands::BrokerId, 693
 - activemq::commands::BrokerInfo, 716
 - activemq::commands::ConnectionControl, 1057
 - activemq::commands::ConnectionError, 1081
 - activemq::commands::ConnectionId, 1107, 1108
 - activemq::commands::ConnectionInfo, 1130
 - activemq::commands::ConsumerControl, 1167
 - activemq::commands::ConsumerId, 1192, 1193
 - activemq::commands::ConsumerInfo, 1217
 - activemq::commands::ControlCommand, 1245
 - activemq::commands::DataArrayResponse, 1269
 - activemq::commands::DataResponse, 1308
 - activemq::commands::DataStructure, 1374
 - activemq::commands::DestinationInfo, 1392
 - activemq::commands::DiscoveryEvent, 1418
 - activemq::commands::ExceptionResponse, 1485
 - activemq::commands::FlushCommand, 1574
 - activemq::commands::IntegerResponse, 1668
 - activemq::commands::JournalQueueAck, 1720
 - activemq::commands::JournalTopicAck, 1743
 - activemq::commands::JournalTrace, 1768
 - activemq::commands::JournalTransaction, 1791
 - activemq::commands::KeepAliveInfo, 1814
 - activemq::commands::LastPartialCommand, 1841
 - activemq::commands::LocalTransactionId, 1876
 - activemq::commands::Message, 2023
 - activemq::commands::MessageAck, 2057
 - activemq::commands::MessageDispatch, 2085
 - activemq::commands::MessageDispatchNotification, 2117
 - activemq::commands::MessageId, 2144
 - activemq::commands::MessagePull, 2204
 - activemq::commands::NetworkBridgeFilter, 2248
 - activemq::commands::PartialCommand, 2320
 - activemq::commands::ProducerAck, 2422
 - activemq::commands::ProducerId, 2448
 - activemq::commands::ProducerInfo, 2471
 - activemq::commands::RemoveInfo, 2538
 - activemq::commands::RemoveSubscriptionInfo, 2562

- activemq::commands::ReplayCommand, 2585
- activemq::commands::Response, 2612
- activemq::commands::SessionId, 2679, 2680
- activemq::commands::SessionInfo, 2703
- activemq::commands::ShutdownInfo, 2758
- activemq::commands::SubscriptionInfo, 2909
- activemq::commands::TransactionId, 3017
- activemq::commands::TransactionInfo, 3038
- activemq::commands::WireFormatInfo, 3156
- activemq::commands::XATransactionId, 3194
- decaf::lang::Boolean, 676
- decaf::lang::Byte, 780
- decaf::lang::Character, 918
- decaf::lang::Comparable, 1010
- decaf::lang::Double, 1446
- decaf::lang::Float, 1548
- decaf::lang::Integer, 1657
- decaf::lang::Long, 1939
- decaf::lang::Short, 2733
- decaf::net::URI, 3101
- decaf::nio::ByteBuffer, 858
- decaf::nio::CharBuffer, 938
- decaf::nio::DoubleBuffer, 1464
- decaf::nio::FloatBuffer, 1567
- decaf::nio::IntBuffer, 1646
- decaf::nio::LongBuffer, 1960
- decaf::nio::ShortBuffer, 2750
- decaf::security::cert::Certificate, 902
- decaf::security::Principal, 2412
- decaf::security_-
 - provider::unix::openssl::OpenSSLX509Principal, 2289
- decaf::security_-
 - provider::unix::openssl::OpenSSLX509Certificate, 2292
- decaf::util::AbstractCollection, 125
- decaf::util::Collection, 987
- decaf::util::concurrent::ConcurrentStlMap, 1031
- decaf::util::concurrent::SynchronousQueue, 2949
- decaf::util::concurrent::TimeUnit, 3008
- decaf::util::Date, 1379
- decaf::util::Map, 1974
- decaf::util::Properties, 2497
- decaf::util::StlList, 2841
- decaf::util::StlMap, 2851
- decaf::util::StlSet, 2870
- decaf::util::UUID, 3144
- Error
 - decaf::util::logging, 117
- error
 - decaf::util::logging::Logger, 1911
 - decaf::util::logging::SimpleLogger, 2785
- ERROR_CMD
 - activemq::wireformat::stomp::StompCommandConstants, 2873
- Exception
 - decaf::lang::Exception, 1478
- exception
 - activemq::commands::ConnectionError, 1083
 - activemq::commands::ExceptionResponse, 1486
- ExceptionResponse
 - activemq::commands::ExceptionResponse, 1485
- ExceptionResponseMarshaller
 - activemq::wireformat::openwire::marshal::v1::ExceptionResponse, 1492
 - activemq::wireformat::openwire::marshal::v2::ExceptionResponse, 1488
 - activemq::wireformat::openwire::marshal::v3::ExceptionResponse, 1496
 - activemq::wireformat::openwire::marshal::v4::ExceptionResponse, 1500
 - activemq::wireformat::openwire::marshal::v5::ExceptionResponse, 1504
- exclusive
 - activemq::commands::ActiveMQDestination, 249
 - activemq::commands::ConsumerInfo, 1222
- execute
 - activemq::cmsutil::CmsTemplate, 973, 974
 - activemq::core::ActiveMQSessionExecutor, 420
 - decaf::util::concurrent::Executor, 1510
 - executeFirst
 - activemq::core::ActiveMQSessionExecutor, 420
- ExecutionException
 - decaf::util::concurrent::ExecutionException, 1507, 1508
- exit
 - activemq::commands::ConnectionControl, 1059
 - decaf::util::logging::Logger, 1911
- expiration
 - activemq::commands::Message, 2035
- failIfReadOnlyBody
 - activemq::commands::ActiveMQMessageTemplate, 332

- failIfReadOnlyProperties
 - activemq::commands::ActiveMQMessageTemplate, 332
- failIfWriteOnlyBody
 - activemq::commands::ActiveMQMessageTemplate, 332
- FailoverTransport
 - activemq::transport::failover::FailoverTransport, 1515
- FailoverTransportListener
 - activemq::transport::failover::FailoverTransportListener, 1522
 - activemq::transport::failover::FailoverTransportListener, 1526
- Fatal
 - decaf::util::logging, 117
- fatal
 - decaf::util::logging::Logger, 1911
 - decaf::util::logging::SimpleLogger, 2785
- faultTolerant
 - activemq::commands::ConnectionControl, 1059
- faultTolerantConfiguration
 - activemq::commands::BrokerInfo, 721
- FileName
 - activemq::commands::BrokerError::StackTraceElement, 2812
- FilterInputStream
 - decaf::io::FilterInputStream, 1530
- FilterOutputStream
 - decaf::io::FilterOutputStream, 1538
- find
 - decaf::internal::util::TimerTaskHeap, 3004
- findFactory
 - activemq::transport::TransportRegistry, 3083
 - activemq::wireformat::WireFormatRegistry, 3184
- fire
 - activemq::core::ActiveMQConnection, 207
 - activemq::core::ActiveMQSession, 413
 - activemq::transport::TransportFilter, 3075
- fireCommand
 - activemq::transport::mock::MockTransport, 2230
- fireException
 - activemq::transport::mock::MockTransport, 2230
- firstMessageId
 - activemq::commands::MessageAck, 2060
- firstNakNumber
 - activemq::commands::ReplayCommand, 2587
- flip
 - decaf::nio::Buffer, 744
 - decaf::lang::Float, 1546
- FLOAT_TYPE
 - activemq::util::PrimitiveValueNode, 2402
- FloatArrayBuffer
 - decaf::internal::nio::FloatArrayBuffer, 1556, 1557
- FloatBuffer
 - decaf::nio::FloatBuffer, 1564
- FloatToIntBits
 - decaf::lang::Float, 1548
- FloatToRawIntBits
 - decaf::lang::Float, 1548
- floatValue
 - activemq::util::PrimitiveValueNode::PrimitiveValue, 2396
 - decaf::lang::Byte, 780
 - decaf::lang::Character, 918
 - decaf::lang::Double, 1446
 - decaf::lang::Float, 1549
 - decaf::lang::Integer, 1658
 - decaf::lang::Long, 1939
 - decaf::lang::Number, 2283
 - decaf::lang::Short, 2733
 - decaf::util::concurrent::atomic::AtomicInteger, 584
- floor
 - decaf::lang::Math, 2003
- flush
 - activemq::commands::ConsumerControl, 1170
 - decaf::internal::io::StandardErrorOutputStream, 2814
 - decaf::internal::io::StandardOutputStream, 2826
 - decaf::io::BufferedOutputStream, 753
 - decaf::io::ByteArrayOutputStream, 838
 - decaf::io::FilterOutputStream, 1539
 - decaf::io::OutputStream, 2317
 - decaf::net::SocketOutputStream, 2805
 - decaf::util::logging::Handler, 1605
 - decaf::util::logging::StreamHandler, 2890
- FlushCommand
 - activemq::commands::FlushCommand, 1574
- FlushCommandMarshaller
 - activemq::wireformat::openwire::marshal::v1::FlushCommand, 1581
 - activemq::wireformat::openwire::marshal::v2::FlushCommand, 1577
 - activemq::wireformat::openwire::marshal::v3::FlushCommand, 1585

- activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller, 585
- activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller, 590
- format
 - decaf::util::logging::Formatter, 1596
 - decaf::util::logging::SimpleFormatter, 2783
- formatId
 - activemq::commands::XATransactionId, 3197
- formatMessage
 - decaf::util::logging::Formatter, 1596
 - decaf::util::logging::SimpleFormatter, 2783
- fromStream
 - activemq::wireformat::stomp::StompFrame, 2876
- fromString
 - decaf::util::UUID, 3144
- front
 - decaf::util::StlQueue, 2861
- FutureResponse
 - activemq::transport::correlator::FutureResponse, 1601
- GeneralSecurityException
 - decaf::security::GeneralSecurityException, 1602, 1603
- generation
 - decaf::util::concurrent::ConditionHandle, 1047
- get
 - decaf::internal::nio::ByteBuffer, 816
 - decaf::internal::nio::CharArrayBuffer, 927
 - decaf::internal::nio::DoubleArrayBuffer, 1457
 - decaf::internal::nio::FloatArrayBuffer, 1559
 - decaf::internal::nio::IntArrayBuffer, 1639
 - decaf::internal::nio::LongArrayBuffer, 1952, 1953
 - decaf::internal::nio::ShortArrayBuffer, 2743
 - decaf::internal::util::ByteArrayAdapter, 791
 - decaf::lang::Pointer, 2347
 - decaf::nio::ByteBuffer, 858, 859
 - decaf::nio::CharBuffer, 938, 939
 - decaf::nio::DoubleBuffer, 1465, 1466
 - decaf::nio::FloatBuffer, 1567, 1568
 - decaf::nio::IntBuffer, 1646–1648
 - decaf::nio::LongBuffer, 1960–1962
 - decaf::nio::ShortBuffer, 2751, 2752
 - decaf::util::concurrent::atomic::AtomicBoolean, 581
 - decaf::util::concurrent::atomic::AtomicInteger, 585
 - decaf::util::concurrent::atomic::AtomicReference, 590
 - decaf::util::concurrent::ConcurrentStlMap, 1031, 1032
 - decaf::util::concurrent::Future, 1599
 - decaf::util::List, 1868
 - decaf::util::Map, 1975, 1976
 - decaf::util::StlList, 2841
 - decaf::util::StlMap, 2851, 2852
- getAckHandler
 - activemq::commands::Message, 2024
- getAckMode
 - activemq::commands::SessionInfo, 2703
- getAcknowledgeMode
 - activemq::cmsutil::PooledSession, 2361
 - activemq::core::ActiveMQSession, 414
 - cms::Session, 2674
- getAckType
 - activemq::commands::MessageAck, 2057
- getAdditionalPredicate
 - activemq::commands::ConsumerInfo, 1217, 1218
- getAlgorithm
 - decaf::security::Key, 1837
- getAndAdd
 - decaf::util::concurrent::atomic::AtomicInteger, 585
- getAndDecrement
 - decaf::util::concurrent::atomic::AtomicInteger, 585
- getAndIncrement
 - decaf::util::concurrent::atomic::AtomicInteger, 585
- getAndSet
 - decaf::util::concurrent::atomic::AtomicBoolean, 581
 - decaf::util::concurrent::atomic::AtomicInteger, 585
 - decaf::util::concurrent::atomic::AtomicReference, 591
- getAnonymousLogger
 - decaf::util::logging::Logger, 1912
- getAprPool
 - decaf::internal::AprPool, 579
- getArrival
 - activemq::commands::Message, 2024
- getAuthority
 - decaf::internal::net::URIType, 3128
 - decaf::net::URI, 3101
- getBacklog
 - decaf::util::concurrent::ThreadPool, 2979
- getBackOffMultiplier

- activemq::transport::failover::FailoverTransport, 1516
- getBackup
 - activemq::transport::failover::BackupTransportPool, 595
- getBackupPoolSize
 - activemq::transport::failover::BackupTransportPool, 595
 - activemq::transport::failover::FailoverTransport, 1516
- getBasicConstraints
 - decaf::security::cert::X509Certificate, 3190
 - decaf::security_ -
 - provider::unix::openssl::OpenSSLX509Certificate, 2292
- getBlockSize
 - decaf::util::concurrent::ThreadPool, 2980
- getBody
 - activemq::wireformat::stomp::StompFrame, 2876
- getBodyBytes
 - activemq::commands::ActiveMQBytesMessage, 171
 - cms::BytesMessage, 877
- getBodyLength
 - activemq::commands::ActiveMQBytesMessage, 171
 - activemq::wireformat::stomp::StompFrame, 2876
 - cms::BytesMessage, 878
- getBool
 - activemq::util::PrimitiveList, 2373
 - activemq::util::PrimitiveMap, 2384
 - activemq::util::PrimitiveValueNode, 2405
- getBoolean
 - activemq::commands::ActiveMQMapMessage, 276
 - cms::MapMessage, 1985
- getBooleanProperty
 - activemq::commands::ActiveMQMessageTemplate, 332
 - activemq::wireformat::openwire::utils::MessagePropertyInterceptor, 2198
 - cms::Message, 2041
- getBranchQualifier
 - activemq::commands::XATransactionId, 3195
- getBrokerId
 - activemq::commands::BrokerInfo, 716, 717
- getBrokerInTime
 - activemq::commands::Message, 2025
- getBrokerName
 - activemq::commands::BrokerInfo, 717
- activemq::commands::DiscoveryEvent, 1418, 1419
- getBrokerOutTime
- activemq::commands::Message, 2025
- getBrokerPath
 - activemq::commands::ConnectionInfo, 1131
 - activemq::commands::ConsumerInfo, 1218
 - activemq::commands::DestinationInfo, 1393
 - activemq::commands::Message, 2025
 - activemq::commands::ProducerInfo, 2472
- getBrokerSequenceId
 - activemq::commands::MessageId, 2144
- getBrokerUploadUrl
 - activemq::commands::BrokerInfo, 717
- getBrokerURL
 - activemq::commands::BrokerInfo, 717
 - activemq::core::ActiveMQConnectionFactory, 215
- getByte
 - activemq::commands::ActiveMQMapMessage, 276
 - activemq::util::PrimitiveList, 2374
 - activemq::util::PrimitiveMap, 2384
 - activemq::util::PrimitiveValueNode, 2405
 - cms::MapMessage, 1985
- getByteArray
 - activemq::util::PrimitiveList, 2374
 - activemq::util::PrimitiveMap, 2384
 - activemq::util::PrimitiveValueNode, 2405
 - decaf::internal::util::ByteArrayAdapter, 792
- getByteProperty
 - activemq::commands::ActiveMQMessageTemplate, 332
 - activemq::wireformat::openwire::utils::MessagePropertyInterceptor, 2198
 - cms::Message, 2041
- getBytes
 - activemq::commands::ActiveMQMapMessage, 277
 - cms::MapMessage, 1985
- getCacheSize
 - activemq::commands::WireFormatInfo, 3156
 - activemq::wireformat::openwire::OpenWireFormat, 2301
- getCapacity
 - decaf::internal::util::ByteArrayAdapter, 792
- getCause
 - activemq::commands::BrokerError, 687
 - cms::CMSException, 962

- decaf::lang::Exception, 1480
- decaf::lang::Throwable, 2985
- getChar
 - activemq::commands::ActiveMQMapMessage, 277
 - activemq::util::PrimitiveList, 2374
 - activemq::util::PrimitiveMap, 2385
 - activemq::util::PrimitiveValueNode, 2405
 - cms::MapMessage, 1986
 - decaf::internal::nio::ByteBuffer, 817
 - decaf::internal::util::ByteArrayAdapter, 792
 - decaf::nio::ByteBuffer, 860
- getCharArray
 - decaf::internal::util::ByteArrayAdapter, 793
- getCharCapacity
 - decaf::internal::util::ByteArrayAdapter, 793
- getClientID
 - activemq::core::ActiveMQConnection, 207
 - cms::Connection, 1054
- getClientId
 - activemq::commands::ActiveMQDestination, 244
 - activemq::commands::ConnectionInfo, 1131
 - activemq::commands::JournalTopicAck, 1743, 1744
 - activemq::commands::RemoveSubscriptionInfo, 2562
 - activemq::commands::SubscriptionInfo, 2909
 - activemq::core::ActiveMQConnectionSupport, 222
- getCloseTimeout
 - activemq::core::ActiveMQConnectionSupport, 222
- getCluster
 - activemq::commands::Message, 2025
- getCMSCorrelationID
 - activemq::commands::ActiveMQMessageTemplate, 333
 - cms::Message, 2042
- getCMSDeliveryMode
 - activemq::commands::ActiveMQMessageTemplate, 333
 - cms::Message, 2042
- getCMSDestination
 - activemq::commands::ActiveMQDestination, 244
 - activemq::commands::ActiveMQMessageTemplate, 333
- activemq::commands::ActiveMQQueue, 381
- activemq::commands::ActiveMQTempQueue, 479
- activemq::commands::ActiveMQTempTopic, 503
- activemq::commands::ActiveMQTopic, 552
- cms::Message, 2042
- getCMSExpiration
 - activemq::commands::ActiveMQMessageTemplate, 333
 - cms::Message, 2043
- getCMSMajorVersion
 - activemq::core::ActiveMQConnectionMetaData, 217
 - cms::ConnectionMetaData, 1155
- getCMSMessageID
 - activemq::commands::ActiveMQMessageTemplate, 334
 - cms::Message, 2043
- getCMSMinorVersion
 - activemq::core::ActiveMQConnectionMetaData, 217
 - cms::ConnectionMetaData, 1156
- getCMSPriority
 - activemq::commands::ActiveMQMessageTemplate, 334
 - cms::Message, 2044
- getCMSProperties
 - activemq::commands::ActiveMQQueue, 381
 - activemq::commands::ActiveMQTempQueue, 480
 - activemq::commands::ActiveMQTempTopic, 503
 - activemq::commands::ActiveMQTopic, 552
 - cms::Destination, 1389
- getCMSProviderName
 - activemq::core::ActiveMQConnectionMetaData, 218
 - cms::ConnectionMetaData, 1156
- getCMSRedelivered
 - activemq::commands::ActiveMQMessageTemplate, 334
 - cms::Message, 2044
- getCMSReplyTo
 - activemq::commands::ActiveMQMessageTemplate, 335
 - cms::Message, 2044
- getCMSTimestamp
 - activemq::commands::ActiveMQMessageTemplate, 335
 - cms::Message, 2045
- getCMSType

- activemq::commands::ActiveMQMessageTemplate, 335
- cms::Message, 2045
- getCMSVersion
 - activemq::core::ActiveMQConnectionMetaData, 218
 - cms::ConnectionMetaData, 1156
- getCMSXPathPropertyNames
 - activemq::core::ActiveMQConnectionMetaData, 218
 - cms::ConnectionMetaData, 1156
- getCommand
 - activemq::commands::ControlCommand, 1245
 - activemq::wireformat::stomp::StompFrame, 2876
- getCommandId
 - activemq::commands::BaseCommand, 599
 - activemq::commands::Command, 992
 - activemq::commands::PartialCommand, 2320
- getCommands
 - activemq::state::TransactionState, 3061
- getComponents
 - activemq::util::CompositeData, 1015
- getConnection
 - activemq::core::ActiveMQSession, 414
- getConnectionFactory
 - activemq::cmsutil::CmsAccessor, 956
- getConnectionId
 - activemq::commands::BrokerInfo, 717
 - activemq::commands::ConnectionError, 1081, 1082
 - activemq::commands::ConnectionInfo, 1131
 - activemq::commands::ConsumerId, 1193
 - activemq::commands::DestinationInfo, 1393
 - activemq::commands::LocalTransactionId, 1877
 - activemq::commands::ProducerId, 2448
 - activemq::commands::RemoveSubscriptionInfo, 2562
 - activemq::commands::SessionId, 2680
 - activemq::commands::TransactionInfo, 3039
 - activemq::core::ActiveMQConnection, 207
- getConnectionInfo
 - activemq::core::ActiveMQConnection, 207
- getConsumerId
 - activemq::commands::ConsumerControl, 1168
 - activemq::commands::ConsumerInfo, 1218
 - activemq::commands::MessageAck, 2057
 - activemq::commands::MessageDispatch, 2086
 - activemq::commands::MessageDispatchNotification, 2118
 - activemq::commands::MessagePull, 2204, 2205
 - activemq::core::ActiveMQConsumer, 234
 - activemq::core::DispatchData, 1440
 - activemq::core::ConsumerInfo
 - activemq::core::ActiveMQConsumer, 235
- getConsumerState
 - activemq::state::SessionState, 2728
- getConsumerStates
 - activemq::state::SessionState, 2728
- getContent
 - activemq::commands::Message, 2025
- getCorrelationId
 - activemq::commands::Message, 2025
 - activemq::commands::MessagePull, 2205
 - activemq::commands::Response, 2613
- getCount
 - decaf::util::concurrent::CountDownLatch, 1267
- getData
 - activemq::commands::DataArrayResponse, 1269, 1270
 - activemq::commands::DataResponse, 1309
 - activemq::commands::PartialCommand, 2321
- getDataStructure
 - activemq::commands::Message, 2025
- getDataStructureType
 - activemq::commands::ActiveMQBlobMessage, 144
 - activemq::commands::ActiveMQBytesMessage, 172
 - activemq::commands::ActiveMQDestination, 244
 - activemq::commands::ActiveMQMapMessage, 277
 - activemq::commands::ActiveMQMessage, 306
 - activemq::commands::ActiveMQObjectMessage, 346
 - activemq::commands::ActiveMQQueue, 381
 - activemq::commands::ActiveMQStreamMessage, 426
 - activemq::commands::ActiveMQTempDestination, 458
 - activemq::commands::ActiveMQTempQueue, 480
 - activemq::commands::ActiveMQTempTopic, 504

- activemq::commands::ActiveMQTextMessage, 528
- activemq::commands::ActiveMQTopic, 552
- activemq::commands::BrokerError, 688
- activemq::commands::BrokerId, 693
- activemq::commands::BrokerInfo, 717
- activemq::commands::ConnectionControl, 1057
- activemq::commands::ConnectionError, 1082
- activemq::commands::ConnectionId, 1108
- activemq::commands::ConnectionInfo, 1131
- activemq::commands::ConsumerControl, 1168
- activemq::commands::ConsumerId, 1193
- activemq::commands::ConsumerInfo, 1218
- activemq::commands::ControlCommand, 1245
- activemq::commands::DataArrayResponse, 1270
- activemq::commands::DataResponse, 1309
- activemq::commands::DataStructure, 1375
- activemq::commands::DestinationInfo, 1393
- activemq::commands::DiscoveryEvent, 1419
- activemq::commands::ExceptionResponse, 1485
- activemq::commands::FlushCommand, 1574
- activemq::commands::IntegerResponse, 1668
- activemq::commands::JournalQueueAck, 1720
- activemq::commands::JournalTopicAck, 1744
- activemq::commands::JournalTrace, 1768
- activemq::commands::JournalTransaction, 1791
- activemq::commands::KeepAliveInfo, 1814
- activemq::commands::LastPartialCommand, 1841
- activemq::commands::LocalTransactionId, 1877
- activemq::commands::Message, 2025
- activemq::commands::MessageAck, 2057
- activemq::commands::MessageDispatch, 2086
- activemq::commands::MessageDispatchNotification, 2118
- activemq::commands::MessageId, 2144
- activemq::commands::MessagePull, 2205
- activemq::commands::NetworkBridgeFilter, 2248
- activemq::commands::PartialCommand, 2321
- activemq::commands::ProducerAck, 2422
- activemq::commands::ProducerId, 2448
- activemq::commands::ProducerInfo, 2472
- activemq::commands::RemoveInfo, 2538
- activemq::commands::RemoveSubscriptionInfo, 2562
- activemq::commands::ReplayCommand, 2586
- activemq::commands::Response, 2613
- activemq::commands::SessionId, 2680
- activemq::commands::SessionInfo, 2704
- activemq::commands::ShutdownInfo, 2758
- activemq::commands::SubscriptionInfo, 2909
- activemq::commands::TransactionId, 3018
- activemq::commands::TransactionInfo, 3039
- activemq::commands::WireFormatInfo, 3156
- activemq::commands::XATransactionId, 3195
- activemq::wireformat::openwire::marshal::DataStreamMarshal, 1336
- activemq::wireformat::openwire::marshal::v1::ActiveMQBlobM, 152
- activemq::wireformat::openwire::marshal::v1::ActiveMQBytes, 187
- activemq::wireformat::openwire::marshal::v1::ActiveMQMapM, 290
- activemq::wireformat::openwire::marshal::v1::ActiveMQMessa, 312
- activemq::wireformat::openwire::marshal::v1::ActiveMQObject, 352
- activemq::wireformat::openwire::marshal::v1::ActiveMQQueue, 388
- activemq::wireformat::openwire::marshal::v1::ActiveMQStream, 441
- activemq::wireformat::openwire::marshal::v1::ActiveMQTemp, 486
- activemq::wireformat::openwire::marshal::v1::ActiveMQTemp, 510
- activemq::wireformat::openwire::marshal::v1::ActiveMQTextM, 535
- activemq::wireformat::openwire::marshal::v1::ActiveMQTopic, 558
- activemq::wireformat::openwire::marshal::v1::BrokerIdMarshal, 699
- activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshal, 727

activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller	2263	activemq::wireformat::openwire::marshal::v1::NetworkBridgeF
1065		
activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller	2336	activemq::wireformat::openwire::marshal::v1::PartialCommand
1088		
activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller	2441	activemq::wireformat::openwire::marshal::v1::ProducerAckMar
1114		
activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller	2463	activemq::wireformat::openwire::marshal::v1::ProducerIdMars
1144		
activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller	2491	activemq::wireformat::openwire::marshal::v1::ProducerInfoMa
1180		
activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller	2542	activemq::wireformat::openwire::marshal::v1::RemoveInfoMars
1204		
activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller	2573	activemq::wireformat::openwire::marshal::v1::RemoveSubscrip
1228		
activemq::wireformat::openwire::marshal::v1::ContainerCommandMarshaller	2604	activemq::wireformat::openwire::marshal::v1::ReplayCommand
1255		
activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller	2634	activemq::wireformat::openwire::marshal::v1::ResponseMarsha
1280		
activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller	2687	activemq::wireformat::openwire::marshal::v1::SessionIdMarsha
1315		
activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller	2710	activemq::wireformat::openwire::marshal::v1::SessionInfoMars
1408		
activemq::wireformat::openwire::marshal::v1::DisconnectInfoMarshaller	2761	activemq::wireformat::openwire::marshal::v1::ShutdownInfoMa
1437		
activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller	2913	activemq::wireformat::openwire::marshal::v1::SubscriptionInfo
1492		
activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller	3050	activemq::wireformat::openwire::marshal::v1::TransactionInfo
1581		
activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller	3172	activemq::wireformat::openwire::marshal::v1::WireFormatInfo
1675		
activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller	3210	activemq::wireformat::openwire::marshal::v1::XATransactionId
1735		
activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller	164	activemq::wireformat::openwire::marshal::v2::ActiveMQBlobM
1752		
activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller	199	activemq::wireformat::openwire::marshal::v2::ActiveMQBytesI
1783		
activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller	302	activemq::wireformat::openwire::marshal::v2::ActiveMQMapM
1806		
activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller	324	activemq::wireformat::openwire::marshal::v2::ActiveMQMessa
1833		
activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller	364	activemq::wireformat::openwire::marshal::v2::ActiveMQObject
1848		
activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller	400	activemq::wireformat::openwire::marshal::v2::ActiveMQQueue
1891		
activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller	453	activemq::wireformat::openwire::marshal::v2::ActiveMQStream
2077		
activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller	498	activemq::wireformat::openwire::marshal::v2::ActiveMQTemp
2109		
activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller	522	activemq::wireformat::openwire::marshal::v2::ActiveMQTemp5
2133		
activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller	547	activemq::wireformat::openwire::marshal::v2::ActiveMQTextM
2163		
activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller	570	activemq::wireformat::openwire::marshal::v2::ActiveMQTopicI
2216		

activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller	711	activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller	2147
activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller	739	activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller	2209
activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller	1077	activemq::wireformat::openwire::marshal::v2::NetworkBridgeFormat	2251
activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller	1100	activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller	2323
activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller	1126	activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller	2425
activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller	1152	activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller	2451
activemq::wireformat::openwire::marshal::v2::ConsumerGroupMarshaller	1188	activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller	2475
activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller	1212	activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller	2549
activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller	1240	activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionMarshaller	2569
activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller	1263	activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller	2589
activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller	1288	activemq::wireformat::openwire::marshal::v2::ResponseMarshaller	2621
activemq::wireformat::openwire::marshal::v2::DataResponseV1Marshaller	1327	activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller	2691
activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller	1397	activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller	2706
activemq::wireformat::openwire::marshal::v2::DiscoveryInfoMarshaller	1421	activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller	2773
activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller	1488	activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller	2928
activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller	1577	activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller	3058
activemq::wireformat::openwire::marshal::v2::IntegrationResponseMarshaller	1671	activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller	3164
activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller	1723	activemq::wireformat::openwire::marshal::v2::XATransactionInfoMarshaller	3198
activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller	1748	activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMarshaller	148
activemq::wireformat::openwire::marshal::v2::JournalTransactionIdMarshaller	1771	activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMarshaller	183
activemq::wireformat::openwire::marshal::v2::JournalTransactionInfoMarshaller	1794	activemq::wireformat::openwire::marshal::v3::ActiveMQMapMarshaller	286
activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller	1817	activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller	308
activemq::wireformat::openwire::marshal::v2::LastReceivedCommandMarshaller	1844	activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMarshaller	348
activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller	1879	activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller	384
activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller	2062	activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMarshaller	437
activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller	2097	activemq::wireformat::openwire::marshal::v3::ActiveMQTempMarshaller	482
activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller	2121	activemq::wireformat::openwire::marshal::v3::ActiveMQTempMarshaller	506

activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller	2105	activemq::wireformat::openwire::marshal::v3::MessageDispatcher
531		
activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMessageMarshaller	2129	activemq::wireformat::openwire::marshal::v3::MessageDispatcher
555		
activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller	2155	activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller
695		
activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller	2212	activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller
723		
activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller	2255	activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilter
1061		
activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller	2328	activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller
1085		
activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller	2429	activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller
1110		
activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller	2455	activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller
1136		
activemq::wireformat::openwire::marshal::v3::ConsumerGroupViewMarshaller	2479	activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller
1172		
activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller	2553	activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller
1196		
activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller	2577	activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionMarshaller
1224		
activemq::wireformat::openwire::marshal::v3::ContactCommandMarshaller	2592	activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller
1248		
activemq::wireformat::openwire::marshal::v3::DataActiveResponseMarshaller	2625	activemq::wireformat::openwire::marshal::v3::ResponseMarshaller
1272		
activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller	2699	activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller
1311		
activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller	2722	activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller
1400		
activemq::wireformat::openwire::marshal::v3::DiscardResponseMarshaller	2769	activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller
1425		
activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller	2916	activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller
1496		
activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller	3046	activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller
1585		
activemq::wireformat::openwire::marshal::v3::IntegrationResponseMarshaller	3180	activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller
1679		
activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller	3202	activemq::wireformat::openwire::marshal::v3::XATransactionInfoMarshaller
1731		
activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller	156	activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMarshaller
1756		
activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller	191	activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMarshaller
1779		
activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller	294	activemq::wireformat::openwire::marshal::v4::ActiveMQMapMarshaller
1798		
activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller	316	activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller
1825		
activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller	356	activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMarshaller
1852		
activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller	392	activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller
1883		
activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller	445	activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMarshaller
2069		

activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller	490	activemq::wireformat::openwire::marshal::v4::LocalTransaction	1887
activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller	514	activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller	2073
activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller	539	activemq::wireformat::openwire::marshal::v4::MessageDispatch	2101
activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMessageMarshaller	562	activemq::wireformat::openwire::marshal::v4::MessageDispatch	2125
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller	703	activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller	2151
activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller	731	activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller	2224
activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller	1069	activemq::wireformat::openwire::marshal::v4::NetworkBridgeF	2267
activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller	1092	activemq::wireformat::openwire::marshal::v4::PartialCommand	2332
activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller	1118	activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller	2433
activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller	1140	activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller	2467
activemq::wireformat::openwire::marshal::v4::ConsumerGroupWithMarshaller	1176	activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller	2487
activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller	1200	activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller	2557
activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller	1232	activemq::wireformat::openwire::marshal::v4::RemoveSubscription	2581
activemq::wireformat::openwire::marshal::v4::ContactCommandMarshaller	1251	activemq::wireformat::openwire::marshal::v4::ReplayCommand	2600
activemq::wireformat::openwire::marshal::v4::DataActiveResponseMarshaller	1276	activemq::wireformat::openwire::marshal::v4::ResponseMarshaller	2639
activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller	1323	activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller	2695
activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller	1404	activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller	2714
activemq::wireformat::openwire::marshal::v4::DiscardInfoMarshaller	1429	activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller	2765
activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller	1500	activemq::wireformat::openwire::marshal::v4::SubscriptionInfo	2924
activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller	1589	activemq::wireformat::openwire::marshal::v4::TransactionInfo	3054
activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller	1683	activemq::wireformat::openwire::marshal::v4::WireFormatInfo	3176
activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller	1727	activemq::wireformat::openwire::marshal::v4::XATransactionId	3206
activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller	1760	activemq::wireformat::openwire::marshal::v5::ActiveMQBlobM	160
activemq::wireformat::openwire::marshal::v4::JournalFrameMarshaller	1775	activemq::wireformat::openwire::marshal::v5::ActiveMQBytesL	195
activemq::wireformat::openwire::marshal::v4::JournalFrameActiveMarshaller	1802	activemq::wireformat::openwire::marshal::v5::ActiveMQMapM	298
activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller	1829	activemq::wireformat::openwire::marshal::v5::ActiveMQMessa	320
activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller	1856	activemq::wireformat::openwire::marshal::v5::ActiveMQObject	360

activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller, openwire::marshal::v5::KeepAliveInfoMarshaller, 396
 activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMessageMarshaller; marshal::v5::LastPartialCommandMarshaller, 449
 activemq::wireformat::openwire::marshal::v5::ActiveMQQueueFormatMarshaller, openwire::marshal::v5::LocalTransactionInfoMarshaller, 494
 activemq::wireformat::openwire::marshal::v5::ActiveMQQueueTopicMarshaller, openwire::marshal::v5::MessageAckMarshaller, 518
 activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMessageMarshaller, openwire::marshal::v5::MessageDispatchMarshaller, 543
 activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMessageMarshaller, openwire::marshal::v5::MessageDispatchMarshaller, 566
 activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller, openwire::marshal::v5::MessageIdMarshaller, 707
 activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller, openwire::marshal::v5::MessagePullMarshaller, 735
 activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller, openwire::marshal::v5::NetworkBridgeFormatMarshaller, 1073
 activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller, openwire::marshal::v5::PartialCommandMarshaller, 1096
 activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller, openwire::marshal::v5::ProducerAckMarshaller, 1122
 activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller, openwire::marshal::v5::ProducerIdMarshaller, 1148
 activemq::wireformat::openwire::marshal::v5::ConsumerGroupMarshaller, openwire::marshal::v5::ProducerInfoMarshaller, 1184
 activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller, openwire::marshal::v5::RemoveInfoMarshaller, 1208
 activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller, openwire::marshal::v5::RemoveSubscriptionMarshaller, 1236
 activemq::wireformat::openwire::marshal::v5::ContainerCommandMarshaller, openwire::marshal::v5::ReplayCommandMarshaller, 1259
 activemq::wireformat::openwire::marshal::v5::DataActiveResponseMarshaller, openwire::marshal::v5::ResponseMarshaller, 1284
 activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller, openwire::marshal::v5::SessionIdMarshaller, 1319
 activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller, openwire::marshal::v5::SessionInfoMarshaller, 1412
 activemq::wireformat::openwire::marshal::v5::DiscardResponseMarshaller, openwire::marshal::v5::ShutdownInfoMarshaller, 1433
 activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller, openwire::marshal::v5::SubscriptionInfoMarshaller, 1504
 activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller, openwire::marshal::v5::TransactionInfoMarshaller, 1593
 activemq::wireformat::openwire::marshal::v5::IntegratedResponseMarshaller, openwire::marshal::v5::WireFormatInfoMarshaller, 1687
 activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller, openwire::marshal::v5::XATransactionInfoMarshaller, 1739
 activemq::wireformat::openwire::marshal::v5::JmsDestinationMarshaller, 1764
 activemq::wireformat::openwire::marshal::v5::JmsDestinationMarshallerName, 1787
 activemq::wireformat::openwire::marshal::v5::JmsDestinationMarshaller, 1810

activemq::cmsutil::CmsTemplate, 974
 activemq::cmsutil::CmsTemplate, 975
 decaf::util::concurrent::Delayed, 1386

- getDeliveryMode
 - activemq::cmsutil::CachedProducer, 892
 - activemq::cmsutil::CmsTemplate, 975
 - activemq::core::ActiveMQProducer, 369
 - cms::MessageProducer, 2191
- getDeliverySequenceId
 - activemq::commands::MessageDispatchNotification, 2118
- getDestination
 - activemq::cmsutil::CmsTemplate::ProducerExecutor, 2445
 - activemq::cmsutil::CmsTemplate::ReceiveExecutor, 2525
 - activemq::cmsutil::CmsTemplate::ResolveProducerExecutor, 2607
 - activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor, 2608
 - activemq::commands::ConsumerInfo, 1218, 1219
 - activemq::commands::DestinationInfo, 1393, 1394
 - activemq::commands::JournalQueueAck, 1720, 1721
 - activemq::commands::JournalTopicAck, 1744, 1745
 - activemq::commands::Message, 2026, 2027
 - activemq::commands::MessageAck, 2057, 2058
 - activemq::commands::MessageDispatch, 2086, 2087
 - activemq::commands::MessageDispatchNotification, 2118
 - activemq::commands::MessagePull, 2205, 2206
 - activemq::commands::ProducerInfo, 2472
 - activemq::commands::SubscriptionInfo, 2909, 2910
- getDestinationResolver
 - activemq::cmsutil::CmsDestinationAccessor, 959
- getDestinationType
 - activemq::commands::ActiveMQDestination, 244
 - activemq::commands::ActiveMQQueue, 382
 - activemq::commands::ActiveMQTempQueue, 480
 - activemq::commands::ActiveMQTempTopic, 504
 - activemq::commands::ActiveMQTopic, 552
 - cms::Destination, 1389
- getDisableMessageID
 - activemq::cmsutil::CachedProducer, 893
 - activemq::core::ActiveMQProducer, 369
 - cms::MessageProducer, 2191
- getDisableMessageTimeStamp
 - activemq::cmsutil::CachedProducer, 893
 - activemq::core::ActiveMQProducer, 369
 - cms::MessageProducer, 2191
- getDouble
 - activemq::commands::ActiveMQMapMessage, 278
 - activemq::util::PrimitiveList, 2375
 - activemq::util::PrimitiveMap, 2385
 - activemq::util::PrimitiveValueNode, 2406
 - cms::MapMessage, 1986
 - decaf::internal::nio::ByteBuffer, 817
 - decaf::internal::util::ByteArrayAdapter, 793
 - decaf::nio::ByteBuffer, 860, 861
- getDoubleArray
 - decaf::internal::util::ByteArrayAdapter, 793
- getDoubleAt
 - decaf::internal::util::ByteArrayAdapter, 794
- getDoubleCapacity
 - decaf::internal::util::ByteArrayAdapter, 794
- getDoubleProperty
 - activemq::commands::ActiveMQMessageTemplate, 335
 - activemq::wireformat::openwire::utils::MessagePropertyInterpreter, 2199
 - cms::Message, 2045
- getEncoded
 - decaf::security::auth::x500::X500Principal, 3189
 - decaf::security::cert::Certificate, 902
 - decaf::security::Key, 1837
 - decaf::security_ -
 - provider::unix::openssl::OpenSSLX500Principal, 2289
 - decaf::security_ -
 - provider::unix::openssl::OpenSSLX509Certificate, 2292
- getEnumeration
 - cms::QueueBrowser, 2512
- getenv
 - decaf::lang::System, 2954
- getErrorCode
 - decaf::net::SocketError, 2792
- getErrorMessage
 - decaf::net::SocketError, 2792
- getException
 - activemq::commands::ConnectionError, 1082

- activemq::commands::ExceptionResponse, 1486
- getExceptionClass
 - activemq::commands::BrokerError, 688
- getExceptionListener
 - activemq::core::ActiveMQConnection, 208
 - activemq::core::ActiveMQSession, 414
 - cms::Connection, 1054
- getExpiration
 - activemq::commands::Message, 2027
- getFilter
 - decaf::util::logging::Handler, 1605
 - decaf::util::logging::Logger, 1912
 - decaf::util::logging::StreamHandler, 2890
- getFirstMessageId
 - activemq::commands::MessageAck, 2058
- getFirstNakNumber
 - activemq::commands::ReplayCommand, 2586
- getFloat
 - activemq::commands::ActiveMQMapMessage, 278
 - activemq::util::PrimitiveList, 2375
 - activemq::util::PrimitiveMap, 2386
 - activemq::util::PrimitiveValueNode, 2406
 - cms::MapMessage, 1986
 - decaf::internal::nio::ByteBuffer, 818
 - decaf::internal::util::ByteArrayAdapter, 794
 - decaf::nio::ByteBuffer, 861
- getFloatArray
 - decaf::internal::util::ByteArrayAdapter, 795
- getFloatAt
 - decaf::internal::util::ByteArrayAdapter, 795
- getFloatCapacity
 - decaf::internal::util::ByteArrayAdapter, 795
- getFloatProperty
 - activemq::commands::ActiveMQMessageTemplate, 336
 - activemq::wireformat::openwire::utils::MessageProperty, 2199
 - cms::Message, 2046
- getFormat
 - decaf::security::Key, 1837
- getFormatId
 - activemq::commands::XATransactionId, 3195
- getFormatter
 - decaf::util::logging::Handler, 1606
 - decaf::util::logging::StreamHandler, 2890
- getFragment
 - activemq::util::CompositeData, 1015
 - decaf::internal::net::URIType, 3128
 - decaf::net::URI, 3101
- getFreeThreadCount
 - decaf::util::concurrent::ThreadPool, 2980
- getGlobalPool
 - decaf::internal::AprPool, 579
 - decaf::internal::DecafRuntime, 1382
- getGlobalTransactionId
 - activemq::commands::XATransactionId, 3196
- getGroupID
 - activemq::commands::Message, 2027
- getGroupSequence
 - activemq::commands::Message, 2027
- getHead
 - decaf::util::logging::Formatter, 1597
 - decaf::util::logging::SimpleFormatter, 2783
- getHoldCount
 - decaf::util::concurrent::locks::ReentrantLock, 2528
- getHost
 - activemq::util::CompositeData, 1015
 - decaf::internal::net::URIType, 3128
 - decaf::net::URI, 3101
- getId
 - activemq::state::TransactionState, 3061
- getIndex
 - decaf::net::URISyntaxException, 3125
- getInfo
 - activemq::state::ConnectionState, 1159
 - activemq::state::ConsumerState, 1243
 - activemq::state::ProducerState, 2494
 - activemq::state::SessionState, 2728
- getInitialDelayTime
 - activemq::transport::inactivity::InactivityMonitor, 1625
- getInitialReconnectDelay
 - activemq::transport::failover::FailoverTransport, 1516
- getInput
 - decaf::net::URISyntaxException, 3125
- getListener
 - decaf::io::Reader, 2519
 - decaf::net::BufferedSocket, 757
 - decaf::net::Socket, 2788
 - decaf::net::TcpSocket, 2962
- getInstance
 - activemq::transport::mock::MockTransport, 2230
 - activemq::transport::TransportRegistry, 3084
 - activemq::wireformat::WireFormatRegistry, 3184

- decaf::security_-
 - provider::SecurityProviderMap, 2648
- decaf::util::concurrent::ThreadPool, 2980
- decaf::util::logging::LogManager, 1926
- decaf::util::logging::LogWriter, 1933
- getInt
 - activemq::commands::ActiveMQMapMessage, 278
 - activemq::util::PrimitiveList, 2376
 - activemq::util::PrimitiveMap, 2386
 - activemq::util::PrimitiveValueNode, 2406
 - cms::MapMessage, 1986
 - decaf::internal::nio::ByteBuffer, 818, 819
 - decaf::internal::util::ByteArrayAdapter, 795
 - decaf::nio::ByteBuffer, 862
- getIntArray
 - decaf::internal::util::ByteArrayAdapter, 796
- getIntAt
 - decaf::internal::util::ByteArrayAdapter, 796
- getIntCapacity
 - decaf::internal::util::ByteArrayAdapter, 796
- getIntProperty
 - activemq::commands::ActiveMQMessageTemplate, 336
 - activemq::wireformat::openwire::utils::MessageProperty, 2199
 - cms::Message, 2046
- getIssuerUniqueID
 - decaf::security::cert::X509Certificate, 3190
 - decaf::security_-
 - provider::unix::openssl::OpenSSLX509Certificate, 2292
- getIssuerX500Principal
 - decaf::security::cert::X509Certificate, 3190
 - decaf::security_-
 - provider::unix::openssl::OpenSSLX509Certificate, 2292
- getKeepAlive
 - decaf::net::BufferedSocket, 757
 - decaf::net::Socket, 2788
 - decaf::net::TcpSocket, 2962
- getKey
 - decaf::util::Map::Entry, 1473
- getKeyUsage
 - decaf::security::cert::X509Certificate, 3191
 - decaf::security_-
 - provider::unix::openssl::OpenSSLX509Certificate, 2293
- getLastDeliveredSequenceId
 - activemq::commands::RemoveInfo, 2539
 - activemq::core::ActiveMQConsumer, 235
 - activemq::core::ActiveMQSession, 414
 - getLastMessageId
 - activemq::commands::MessageAck, 2058
 - getLastNakNumber
 - activemq::commands::ReplayCommand, 2586
 - getLastSequenceId
 - activemq::util::LongSequenceGenerator, 1967
 - getLeastSignificantBits
 - decaf::util::UUID, 3144
 - getLevel
 - decaf::util::logging::Handler, 1606
 - decaf::util::logging::Logger, 1912
 - decaf::util::logging::LogRecord, 1929
 - decaf::util::logging::StreamHandler, 2890
 - getLimit
 - activemq::util::MemoryUsage, 2016
 - getList
 - activemq::util::PrimitiveValueNode, 2406
 - getLogger
 - decaf::util::logging::Logger, 1912
 - decaf::util::logging::LogManager, 1926
 - getLoggerName
 - decaf::util::logging::LogRecord, 1929
 - getLoggerNames
 - decaf::util::logging::LogManager, 1926
 - getProperty
 - activemq::commands::ActiveMQMapMessage, 279
 - activemq::util::PrimitiveList, 2376
 - activemq::util::PrimitiveMap, 2386
 - activemq::util::PrimitiveValueNode, 2407
 - cms::MapMessage, 1987
 - decaf::internal::nio::ByteBuffer, 819
 - decaf::internal::util::ByteArrayAdapter, 797
 - decaf::nio::ByteBuffer, 862, 863
 - decaf::util::LongArray, 797
 - decaf::internal::util::ByteArrayAdapter, 797
 - getLongAt
 - decaf::internal::util::ByteArrayAdapter, 797
 - getLongCapacity
 - decaf::internal::util::ByteArrayAdapter, 798
 - getLongProperty
 - activemq::commands::ActiveMQMessageTemplate, 337
 - activemq::wireformat::openwire::utils::MessagePropertyInterceptor, 2199

- cms::Message, 2046
- getMagic
 - activemq::commands::WireFormatInfo, 3157
- getMap
 - activemq::commands::ActiveMQMapMessage, 279
 - activemq::util::PrimitiveValueNode, 2407
- getMapNames
 - activemq::commands::ActiveMQMapMessage, 279
 - cms::MapMessage, 1987
- getMarshaledForm
 - activemq::commands::BaseDataStructure, 661
 - activemq::wireformat::MarshalAware, 1994
- getMarshaledProperties
 - activemq::commands::Message, 2027
 - activemq::commands::WireFormatInfo, 3157
- getMaxCacheSize
 - activemq::state::ConnectionStateTracker, 1162
 - activemq::transport::failover::FailoverTransport, 1516
- getMaximumPendingMessageLimit
 - activemq::commands::ConsumerInfo, 1219
- getMaximumRedeliveries
 - activemq::core::ActiveMQTransactionContext, 575
- getMaxInactivityDuration
 - activemq::commands::WireFormatInfo, 3157
 - activemq::wireformat::openwire::OpenWireFormat, 2301
- getMaxInactivityDurationInitalDelay
 - activemq::commands::WireFormatInfo, 3157
- getMaxInactivityDurationInitialDelay
 - activemq::wireformat::openwire::OpenWireFormat, 2301
- getMaxReconnectAttempts
 - activemq::transport::failover::FailoverTransport, 1516
- getMaxReconnectDelay
 - activemq::transport::failover::FailoverTransport, 1516
- getMaxThreads
 - decaf::util::concurrent::ThreadPool, 2980
- getMessage
 - activemq::cmsutil::CmsTemplate::ReceiveExecutor, 2526
 - activemq::commands::BrokerError, 688
 - activemq::commands::JournalTrace, 1768, 1769
 - activemq::commands::MessageDispatch, 2087
 - activemq::core::DispatchData, 1440
 - cms::CMSException, 962
 - decaf::lang::Exception, 1480
 - decaf::lang::Throwable, 2985
 - decaf::util::logging::LogRecord, 1929
 - getMessageAck
 - activemq::commands::JournalQueueAck, 1721
 - getMessageCount
 - activemq::commands::MessageAck, 2058
 - getMessageId
 - activemq::commands::JournalTopicAck, 1745
 - activemq::commands::Message, 2027
 - activemq::commands::MessageDispatchNotification, 2118
 - activemq::commands::MessagePull, 2206
 - getMessageListener
 - activemq::cmsutil::CachedConsumer, 889
 - activemq::core::ActiveMQConsumer, 235
 - cms::MessageConsumer, 2081
 - getMessageProperties
 - activemq::commands::Message, 2027
 - getMessageSelector
 - activemq::cmsutil::CachedConsumer, 889
 - activemq::core::ActiveMQConsumer, 235
 - cms::MessageConsumer, 2081
 - cms::QueueBrowser, 2512
 - getMessageSequenceId
 - activemq::commands::JournalTopicAck, 1745
 - getMetaData
 - activemq::core::ActiveMQConnection, 208
 - cms::Connection, 1054
 - getMimeType
 - activemq::commands::ActiveMQBlobMessage, 144
 - getMostSignificantBits
 - decaf::util::UUID, 3144
 - getName
 - activemq::commands::ActiveMQBlobMessage, 144
 - decaf::security::auth::x500::X500Principal, 3189
 - decaf::security::Principal, 2412
 - decaf::security_-
 - provider::unix::openssl::OpenSSLX500Principal, 2289
 - decaf::util::logging::Logger, 1913
 - getNetworkBrokerId

- activemq::commands::NetworkBridgeFilter, 2248, 2249
- getNetworkConsumerPath
 - activemq::commands::ConsumerInfo, 1219
- getNetworkProperties
 - activemq::commands::BrokerInfo, 717, 718
- getNetworkTTL
 - activemq::commands::NetworkBridgeFilter, 2249
- getNextLocalTransactionId
 - activemq::core::ActiveMQConnectionSupport, 222
- getNextSequenceId
 - activemq::util::LongSequenceGenerator, 1967
- getNextSessionId
 - activemq::core::ActiveMQConnectionSupport, 223
- getNextTempDestinationId
 - activemq::core::ActiveMQConnectionSupport, 223
- getNotAfter
 - decaf::security::cert::X509Certificate, 3191
 - decaf::security__ -
 - provider::unix::openssl::OpenSSLX509Certificate, 1131, 1132
- getNotBefore
 - decaf::security::cert::X509Certificate, 3191
 - decaf::security__ -
 - provider::unix::openssl::OpenSSLX509Certificate, 1131, 1132
- getNumReceivedMessageBeforeFail
 - activemq::transport::mock::MockTransport, 2231
- getNumReceivedMessages
 - activemq::transport::mock::MockTransport, 2231
- getNumSentKeepAlives
 - activemq::transport::mock::MockTransport, 2231
- getNumSentKeepAlivesBeforeFail
 - activemq::transport::mock::MockTransport, 2231
- getNumSentMessageBeforeFail
 - activemq::transport::mock::MockTransport, 2231
- getNumSentMessages
 - activemq::transport::mock::MockTransport, 2231
- getObjectId
 - activemq::commands::RemoveInfo, 2539
- getOperationType
 - activemq::commands::DestinationInfo, 1394
- getOptions
 - activemq::commands::ActiveMQDestination, 244
- getOrderedTarget
 - activemq::commands::ActiveMQDestination, 245
- getOriginalDestination
 - activemq::commands::Message, 2027, 2028
- getOriginalTransactionId
 - activemq::commands::Message, 2028
- getOutputStream
 - decaf::io::Writer, 3187
 - decaf::net::BufferedSocket, 758
 - decaf::net::Socket, 2788
 - decaf::net::TcpSocket, 2963
 - decaf::util::logging::StreamHandler, 2890
- getParameters
 - activemq::util::CompositeData, 1015
- getParentId
 - activemq::commands::ConsumerId, 1193
 - activemq::commands::ProducerId, 2449
 - activemq::commands::SessionId, 2680
- getPassword
 - activemq::commands::ConnectionInfo, 215
 - activemq::core::ActiveMQConnectionFactory, 215
 - activemq::core::ActiveMQConnectionSupport, 223
- getPeerBrokerInfos
 - activemq::commands::BrokerInfo, 718
- getPhysicalName
 - activemq::commands::ActiveMQDestination, 245
- getPooledThreadListener
 - decaf::util::concurrent::PooledThread, 2365
- getPoolSize
 - decaf::util::concurrent::ThreadPool, 2980
- getPort
 - decaf::internal::net::URIType, 3129
 - decaf::net::URI, 3102
- getPreferredWireFormatInfo
 - activemq::wireformat::openwire::OpenWireFormat, 2301
- getPrefetch
 - activemq::commands::ConsumerControl, 1168
- getPrefetchSize
 - activemq::commands::ConsumerInfo, 1219
- getPreparedResult

- activemq::state::TransactionState, 3061
- getPriority
 - activemq::cmsutil::CachedProducer, 893
 - activemq::cmsutil::CmsTemplate, 975
 - activemq::commands::ConsumerInfo, 1219
 - activemq::commands::Message, 2028
 - activemq::core::ActiveMQProducer, 370
 - cms::MessageProducer, 2192
- getProducerId
 - activemq::commands::Message, 2028
 - activemq::commands::MessageId, 2144, 2145
 - activemq::commands::ProducerAck, 2422
 - activemq::commands::ProducerInfo, 2472
 - activemq::core::ActiveMQProducer, 370
- getProducerInfo
 - activemq::core::ActiveMQProducer, 370
- getProducerSequenceId
 - activemq::commands::MessageId, 2145
- getProducerState
 - activemq::state::SessionState, 2728
- getProducerStates
 - activemq::state::SessionState, 2728
- getProducerWindowSize
 - activemq::core::ActiveMQConnectionSupport, 223
- getProperties
 - activemq::commands::WireFormatInfo, 3157, 3158
 - activemq::core::ActiveMQConnectionSupport, 223
 - activemq::util::ActiveMQProperties, 376, 377
 - activemq::wireformat::stomp::StompFrame, 2876, 2877
 - decaf::util::logging::LogManager, 1926
- getProperty
 - activemq::util::ActiveMQProperties, 377
 - activemq::wireformat::stomp::StompFrame, 2877
 - cms::CMSProperties, 966
 - decaf::util::logging::LogManager, 1926
 - decaf::util::Properties, 2497
- getPropertyNames
 - activemq::commands::ActiveMQMessageTemplate, 337
 - cms::Message, 2047
- getProvider
 - decaf::security_-
 - provider::SecurityProviderRegistrar, 2650
- getProviderMajorVersion
 - activemq::core::ActiveMQConnectionMetaData, 219
- cms::ConnectionMetaData, 1157
- getProviderMinorVersion
 - activemq::core::ActiveMQConnectionMetaData, 219
 - cms::ConnectionMetaData, 1157
- getProviderVersion
 - activemq::core::ActiveMQConnectionMetaData, 219
 - cms::ConnectionMetaData, 1157
- getPublicKey
 - decaf::security::cert::Certificate, 902, 903
 - decaf::security_-
 - provider::unix::openssl::OpenSSLX509Certificate, 2293
- getQuery
 - decaf::internal::net::URIType, 3129
 - decaf::net::URI, 3102
- getQueue
 - cms::QueueBrowser, 2512
- getQueueName
 - activemq::commands::ActiveMQQueue, 382
 - activemq::commands::ActiveMQTempQueue, 480
 - cms::Queue, 2511
 - cms::TemporaryQueue, 2972
- getRawAuthority
 - decaf::net::URI, 3102
- getRawFragment
 - decaf::net::URI, 3102
- getRawPath
 - decaf::net::URI, 3102
- getRawQuery
 - decaf::net::URI, 3102
- getRawSchemeSpecificPart
 - decaf::net::URI, 3103
- getRawUserInfo
 - decaf::net::URI, 3103
- getReadCheckTime
 - activemq::transport::inactivity::InactivityMonitor, 1626
- getReason
 - decaf::net::URISyntaxException, 3125
- getReceiveBufferSize
 - decaf::net::BufferedSocket, 758
 - decaf::net::Socket, 2788
 - decaf::net::TcpSocket, 2963
- getReceiveTimeout
 - activemq::cmsutil::CmsTemplate, 975
- getReconnectDelay
 - activemq::transport::failover::FailoverTransport, 1516
- getRedeliveryCounter
 - activemq::commands::Message, 2028

- activemq::commands::MessageDispatch, 2087
- getRedeliveryDelay
 - activemq::core::ActiveMQTransactionContext, 575
- getReferences
 - decaf::internal::nio::ByteArrayPerspective, 847
- getRemoteAddress
 - activemq::transport::failover::FailoverTransport, 1516
 - activemq::transport::IOTransport, 1711
 - activemq::transport::mock::MockTransport, 2231
 - activemq::transport::Transport, 3067
 - activemq::transport::TransportFilter, 3076
- getRemoteBlobUrl
 - activemq::commands::ActiveMQBlobMessage, 144
- getReplyTo
 - activemq::commands::Message, 2028
- getResourceLifecycleManager
 - activemq::cmsutil::CmsAccessor, 956
 - activemq::cmsutil::SessionPool, 2726
- getResponse
 - activemq::transport::correlator::FutureResponse, 1601
- getResult
 - activemq::commands::IntegerResponse, 1669
- getReuseAddress
 - decaf::net::BufferedSocket, 758
 - decaf::net::Socket, 2789
 - decaf::net::TcpSocket, 2963
- getRuntime
 - decaf::lang::Runtime, 2643
- getSafeValue
 - decaf::util::StlQueue, 2861
- getScheme
 - activemq::util::CompositeData, 1015
 - decaf::internal::net::URIType, 3129
 - decaf::net::URI, 3103
- getSchemeSpecificPart
 - decaf::internal::net::URIType, 3129
 - decaf::net::URI, 3103
- getSecurityProviderNames
 - decaf::security_-
 - provider::SecurityProviderMap, 2648
- getSelector
 - activemq::commands::ConsumerInfo, 1219
 - activemq::commands::SubscriptionInfo, 2910
- getSendBufferSize
 - decaf::net::BufferedSocket, 758
- decaf::net::Socket, 2789
- decaf::net::TcpSocket, 2963
- getSendTimeout
 - activemq::core::ActiveMQConnectionSupport, 224
 - activemq::core::ActiveMQProducer, 370
- getServiceName
 - activemq::commands::DiscoveryEvent, 1419
- getSession
 - activemq::cmsutil::PooledSession, 2362
- getSessionAcknowledgeMode
 - activemq::cmsutil::CmsAccessor, 956
- getSessionId
 - activemq::commands::ConsumerId, 1194
 - activemq::commands::ProducerId, 2449
 - activemq::commands::SessionInfo, 2704
 - activemq::core::ActiveMQSession, 414
- getSessionInfo
 - activemq::core::ActiveMQSession, 414
- getSessionState
 - activemq::state::ConnectionState, 1159
- getSessionStates
 - activemq::state::ConnectionState, 1159
- getShort
 - activemq::commands::ActiveMQMapMessage, 279
 - activemq::util::PrimitiveList, 2376
 - activemq::util::PrimitiveMap, 2387
 - activemq::util::PrimitiveValueNode, 2407
 - cms::MapMessage, 1987
 - decaf::internal::nio::ByteBuffer, 820
 - decaf::internal::util::ByteArrayAdapter, 798
 - decaf::nio::ByteBuffer, 863
- getShortArray
 - decaf::internal::util::ByteArrayAdapter, 798
- getShortAt
 - decaf::internal::util::ByteArrayAdapter, 798
- getShortCapacity
 - decaf::internal::util::ByteArrayAdapter, 799
- getShortProperty
 - activemq::commands::ActiveMQMessageTemplate, 337
 - activemq::wireformat::openwire::utils::MessagePropertyInterce, 2200
 - cms::Message, 2047
- getSigAlgName
 - decaf::security::cert::X509Certificate, 3191
 - decaf::security_-
 - provider::unix::openssl::OpenSSLX509Certificate,

- 2293
- getSigAlgOID
 - decaf::security::cert::X509Certificate, 3191
 - decaf::security_ -
 - provider::unix::openssl::OpenSSLX509Certificate, 337
 - 2294
- getSigAlgParams
 - decaf::security::cert::X509Certificate, 3191
 - decaf::security_ -
 - provider::unix::openssl::OpenSSLX509Certificate, 2294
- getSignature
 - decaf::security::cert::X509Certificate, 3191
 - decaf::security_ -
 - provider::unix::openssl::OpenSSLX509Certificate, 2294
- getSize
 - activemq::commands::ActiveMQTextMessage, 528
 - activemq::commands::Message, 2028
 - activemq::commands::ProducerAck, 2422
- getSocketHandle
 - decaf::net::TcpSocket, 2964
- getSoLinger
 - decaf::net::BufferedSocket, 759
 - decaf::net::Socket, 2789
 - decaf::net::TcpSocket, 2964
- getSoTimeout
 - decaf::net::BufferedSocket, 759
 - decaf::net::Socket, 2789
 - decaf::net::TcpSocket, 2964
- getSource
 - decaf::internal::net::URIType, 3129
- getSourceFile
 - decaf::util::logging::LogRecord, 1929
- getSourceFunction
 - decaf::util::logging::LogRecord, 1930
- getSourceLine
 - decaf::util::logging::LogRecord, 1930
- getStackTrace
 - cms::CMSException, 962
 - decaf::lang::Exception, 1480
 - decaf::lang::Throwable, 2985
- getStackTraceElements
 - activemq::commands::BrokerError, 688
- getStackTraceString
 - cms::CMSException, 962
 - decaf::lang::Exception, 1480
 - decaf::lang::Throwable, 2986
- getString
 - activemq::commands::ActiveMQMapMessage, 280
 - activemq::util::PrimitiveList, 2377
 - activemq::util::PrimitiveMap, 2387
 - activemq::util::PrimitiveValueNode, 2407
 - cms::MapMessage, 1988
- getStringProperty
 - activemq::commands::ActiveMQMessageTemplate, 2200
 - activemq::wireformat::openwire::utils::MessagePropertyInterce, 2047
 - cms::Message, 2047
- getSubscriptionName
 - activemq::commands::RemoveSubscriptionInfo, 2562, 2563
 - activemq::commands::SubscriptionInfo, 2910
- getSubjectUniqueID
 - decaf::security::cert::X509Certificate, 3191
 - decaf::security_ -
 - provider::unix::openssl::OpenSSLX509Certificate, 2294
- getSubjectX500Principal
 - decaf::security::cert::X509Certificate, 3192
 - decaf::security_ -
 - provider::unix::openssl::OpenSSLX509Certificate, 2294
- getSubscribedDestination
 - activemq::commands::SubscriptionInfo, 2910
- getSubscriptionName
 - activemq::commands::ConsumerInfo, 1219
- getSubscriptionName
 - activemq::commands::JournalTopicAck, 1745
- getTail
 - decaf::util::logging::Formatter, 1597
 - decaf::util::logging::SimpleFormatter, 2783
- getTargetConsumerId
 - activemq::commands::Message, 2028, 2029
- getTBSCertificate
 - decaf::security::cert::X509Certificate, 3192
 - decaf::security_ -
 - provider::unix::openssl::OpenSSLX509Certificate, 2294
- getTcpNoDelay
 - decaf::net::TcpSocket, 2964
- getTempDesinations
 - activemq::state::ConnectionState, 1159
- getText
 - activemq::commands::ActiveMQTextMessage, 528
 - cms::TextMessage, 2975
- getTime
 - decaf::util::Date, 1380
- getTimeout
 - activemq::commands::DestinationInfo, 1394

- activemq::commands::MessagePull, 2206
- activemq::transport::failover::FailoverTransport, 1516
- getTimestamp
 - activemq::commands::Message, 2029
 - decaf::util::logging::LogRecord, 1930
- getTimeToLive
 - activemq::cmsutil::CachedProducer, 893
 - activemq::cmsutil::CmsTemplate, 975
 - activemq::core::ActiveMQProducer, 370
 - cms::MessageProducer, 2192
- getTopicName
 - activemq::commands::ActiveMQTempTopic, 504
 - activemq::commands::ActiveMQTopic, 552
 - cms::TemporaryTopic, 2973
 - cms::Topic, 3014
- getTransactionId
 - activemq::commands::JournalTopicAck, 1745
 - activemq::commands::JournalTransaction, 1792
 - activemq::commands::Message, 2029
 - activemq::commands::MessageAck, 2058
 - activemq::commands::TransactionInfo, 3039
 - activemq::core::ActiveMQTransactionContext, 575
- getTransactionState
 - activemq::state::ConnectionState, 1159
- getTransactionStates
 - activemq::state::ConnectionState, 1159
- getTransport
 - activemq::core::ActiveMQConnectionSupport, 224
 - activemq::transport::failover::BackupTransport, 592
- getTransportListener
 - activemq::transport::failover::FailoverTransport, 1516
 - activemq::transport::IOTransport, 1712
 - activemq::transport::mock::MockTransport, 2231
 - activemq::transport::Transport, 3067
 - activemq::transport::TransportFilter, 3076
- getTransportNames
 - activemq::transport::TransportRegistry, 3084
- getTreadId
 - decaf::util::logging::LogRecord, 1930
- getType
 - activemq::commands::JournalTransaction, 1792
 - activemq::commands::Message, 2029
- activemq::commands::TransactionInfo, 3039
- activemq::util::PrimitiveValueNode, 2408
- decaf::security::cert::Certificate, 903
- decaf::security_ -
 - provider::unix::openssl::OpenSSLX509Certificate, 2294
- getUnconsumedMessages
 - activemq::core::ActiveMQSessionExecutor, 420
- getURI
 - activemq::transport::failover::URIPool, 3118
- getUri
 - activemq::transport::failover::BackupTransport, 592
- getUsage
 - activemq::util::MemoryUsage, 2016
- getUseParentHandlers
 - decaf::util::logging::Logger, 1913
- getUserID
 - activemq::commands::Message, 2029
- getUserInfo
 - decaf::internal::net::URIType, 3129
 - decaf::net::URI, 3103
- getUserName
 - activemq::commands::ConnectionInfo, 1132
- getUsername
 - activemq::core::ActiveMQConnectionFactory, 215
 - activemq::core::ActiveMQConnectionSupport, 224
- getValue
 - activemq::commands::BrokerId, 693
 - activemq::commands::ConnectionId, 1108
 - activemq::commands::ConsumerId, 1194
 - activemq::commands::LocalTransactionId, 1877
 - activemq::commands::ProducerId, 2449
 - activemq::commands::SessionId, 2680
 - activemq::util::PrimitiveValueNode, 2408
 - decaf::util::Map::Entry, 1473
- getVersion
 - activemq::commands::WireFormatInfo, 3158
 - activemq::wireformat::openwire::OpenWireFormat, 2301
 - activemq::wireformat::stomp::StompWireFormat, 2884
 - activemq::wireformat::WireFormat, 3149
 - decaf::security::cert::X509Certificate, 3192
 - decaf::security_ -
 - provider::unix::openssl::OpenSSLX509Certificate,

- 2295
- getWasPrepared
 - activemq::commands::JournalTransaction, 1792
- getWhen
 - decaf::util::TimerTask, 3002
- getWindowSize
 - activemq::commands::ProducerInfo, 2472
- getWireFormat
 - activemq::transport::mock::MockTransport, 2231
- getWireFormatNames
 - activemq::wireformat::WireFormatRegistry, 3185
- getWriteCheckTime
 - activemq::transport::inactivity::InactivityMonitor, 1626
- getX509Name
 - decaf::security_ -
 - provider::unix::openssl::OpenSSLX500Principal, 2289
- globalTransactionId
 - activemq::commands::XATransactionId, 3197
- groupID
 - activemq::commands::Message, 2035
- groupSequence
 - activemq::commands::Message, 2035
- handleTransportFailure
 - activemq::transport::failover::FailoverTransport, 1516
- hasArray
 - decaf::internal::nio::ByteBuffer, 820
 - decaf::internal::nio::CharArrayBuffer, 927
 - decaf::internal::nio::DoubleArrayBuffer, 1457
 - decaf::internal::nio::FloatArrayBuffer, 1560
 - decaf::internal::nio::IntArrayBuffer, 1639
 - decaf::internal::nio::LongArrayBuffer, 1953
 - decaf::internal::nio::ShortArrayBuffer, 2743
 - decaf::nio::ByteBuffer, 864
 - decaf::nio::CharBuffer, 939
 - decaf::nio::DoubleBuffer, 1466
 - decaf::nio::FloatBuffer, 1568
 - decaf::nio::IntBuffer, 1648
 - decaf::nio::LongBuffer, 1962
 - decaf::nio::ShortBuffer, 2752
- hashCode
 - decaf::security::auth::x500::X500Principal, 3189
- hasMoreTokens
 - decaf::util::StringTokenizer, 2905
- hasNegotiator
 - activemq::wireformat::openwire::OpenWireFormat, 2302
 - activemq::wireformat::stomp::StompWireFormat, 2884
 - activemq::wireformat::WireFormat, 3149
- hasNext
 - decaf::util::Iterator, 1717
- hasPrevious
 - decaf::util::ListIterator, 1873
- hasProperty
 - activemq::util::ActiveMQProperties, 377
 - activemq::wireformat::stomp::StompFrame, 2877
 - cms::CMSProperties, 967
 - decaf::util::Properties, 2498
- hasRemaining
 - decaf::nio::Buffer, 745
- hasUnconsumedMessages
 - activemq::core::ActiveMQSessionExecutor, 420
- HAVE_PTHREAD_H
 - activemq/util/Config.h, 3306
 - decaf/util/Config.h, 3306
- HAVE_UUID_T
 - activemq/util/Config.h, 3306
 - decaf/util/Config.h, 3306
- HAVE_UUID_UUID_H
 - activemq/util/Config.h, 3306
 - decaf/util/Config.h, 3306
- HEADER_ACK
 - activemq::wireformat::stomp::StompCommandConstants, 2873
- HEADER_CLIENT_ID
 - activemq::wireformat::stomp::StompCommandConstants, 2873
- HEADER_CONSUMERPRIORITY
 - activemq::wireformat::stomp::StompCommandConstants, 2873
- HEADER_CONTENTLENGTH
 - activemq::wireformat::stomp::StompCommandConstants, 2873
- HEADER_CORRELATIONID
 - activemq::wireformat::stomp::StompCommandConstants, 2873
- HEADER_DESTINATION
 - activemq::wireformat::stomp::StompCommandConstants, 2873
- HEADER_DISPATCH_ASYNC
 - activemq::wireformat::stomp::StompCommandConstants, 2873
- HEADER_EXCLUSIVE
 - activemq::wireformat::stomp::StompCommandConstants, 2873

- HEADER_EXPIRES
 activemq::wireformat::stomp::StompCommandConstants, 2873
- HEADER_ID
 activemq::wireformat::stomp::StompCommandConstants, 2873
- HEADER_JMSPRIORITY
 activemq::wireformat::stomp::StompCommandConstants, 2873
- HEADER_LOGIN
 activemq::wireformat::stomp::StompCommandConstants, 2873
- HEADER_MAXPENDINGMSGLIMIT
 activemq::wireformat::stomp::StompCommandConstants, 2873
- HEADER_MESSAGE
 activemq::wireformat::stomp::StompCommandConstants, 2873
- HEADER_MESSAGEID
 activemq::wireformat::stomp::StompCommandConstants, 2873
- HEADER_NOLOCAL
 activemq::wireformat::stomp::StompCommandConstants, 2873
- HEADER_OLDSUBSCRIPTIONNAME
 activemq::wireformat::stomp::StompCommandConstants, 2873
- HEADER_PASSWORD
 activemq::wireformat::stomp::StompCommandConstants, 2873
- HEADER_PERSISTENT
 activemq::wireformat::stomp::StompCommandConstants, 2873
- HEADER_PREFETCHSIZE
 activemq::wireformat::stomp::StompCommandConstants, 2873
- HEADER_RECEIPT_REQUIRED
 activemq::wireformat::stomp::StompCommandConstants, 2873
- HEADER_RECEIPTID
 activemq::wireformat::stomp::StompCommandConstants, 2873
- HEADER_REDELIVERED
 activemq::wireformat::stomp::StompCommandConstants, 2873
- HEADER_REDELIVERYCOUNT
 activemq::wireformat::stomp::StompCommandConstants, 2873
- HEADER_REPLYTO
 activemq::wireformat::stomp::StompCommandConstants, 2873
- HEADER_REQUESTID
 activemq::wireformat::stomp::StompCommandConstants, 2873
- HEADER_RESPONSEID
 activemq::wireformat::stomp::StompCommandConstants, 2873
- HEADER_RETROACTIVE
 activemq::wireformat::stomp::StompCommandConstants, 2873
- HEADER_SELECTOR
 activemq::wireformat::stomp::StompCommandConstants, 2873
- HEADER_SESSIONID
 activemq::wireformat::stomp::StompCommandConstants, 2873
- HEADER_SUBSCRIPTION
 activemq::wireformat::stomp::StompCommandConstants, 2873
- HEADER_SUBSCRIPTIONNAME
 activemq::wireformat::stomp::StompCommandConstants, 2873
- HEADER_TIMESTAMP
 activemq::wireformat::stomp::StompCommandConstants, 2873
- HEADER_TRANSACTIONID
 activemq::wireformat::stomp::StompCommandConstants, 2873
- HEADER_TRANSFORMATION
 activemq::wireformat::stomp::StompCommandConstants, 2873
- HEADER_TRANSFORMATION_ERROR
 activemq::wireformat::stomp::StompCommandConstants, 2873
- HEADER_TYPE
 activemq::wireformat::stomp::StompCommandConstants, 2873
- HexStringParser
 default::internal::util::HexStringParser, 1608
- HexTable
 activemq::wireformat::openwire::utils::HexTable, 1610
- highestOneBit
 decaf::lang::Integer, 1658
- HOURS
 decaf::util::concurrent::TimeUnit, 3013
- HttpRetryException
 decaf::net::HttpRetryException, 1611, 1612
- ID_ACTIVEMQBLOBMESSAGE
 activemq::commands::ActiveMQBlobMessage, 146
- ID_ACTIVEMQBYTESMESSAGE
 activemq::commands::ActiveMQBytesMessage, 182
- ID_ACTIVEMQDESTINATION

activemq::commands::ActiveMQDestination, 249	activemq::commands::ControlCommand, 1246
ID_ ACTIVEMQMAPMESSAGE	ID_ DATAARRAYRESPONSE
activemq::commands::ActiveMQMapMessage, 284	activemq::commands::DataArrayResponse, 1270
ID_ ACTIVEMQMESSAGE	ID_ DATARESPONSE
activemq::commands::ActiveMQMessage, 307	activemq::commands::DataResponse, 1310
ID_ ACTIVEMQOBJECTMESSAGE	ID_ DESTINATIONINFO
activemq::commands::ActiveMQObjectMessage, 347	activemq::commands::DestinationInfo, 1395
ID_ ACTIVEMQQUEUE	ID_ DISCOVERYEVENT
activemq::commands::ActiveMQQueue, 382	activemq::commands::DiscoveryEvent, 1420
ID_ ACTIVEMQSTREAMMESSAGE	ID_ EXCEPTIONRESPONSE
activemq::commands::ActiveMQStreamMessage, 435	activemq::commands::ExceptionResponse, 1486
ID_ ACTIVEMQTEMPDESTINATION	ID_ FLUSHCOMMAND
activemq::commands::ActiveMQTempDestination, 458	activemq::commands::FlushCommand, 1575
ID_ ACTIVEMQTEMPQUEUE	ID_ INTEGERRESPONSE
activemq::commands::ActiveMQTempQueue, 481	activemq::commands::IntegerResponse, 1669
ID_ ACTIVEMQTEMPTOPIC	ID_ JOURNALQUEUEACK
activemq::commands::ActiveMQTempTopic, 505	activemq::commands::JournalQueueAck, 1721
ID_ ACTIVEMQTEXTMESSAGE	ID_ JOURNALTOPICACK
activemq::commands::ActiveMQTextMessage, 529	activemq::commands::JournalTopicAck, 1746
ID_ ACTIVEMQTOPIC	ID_ JOURNALTRACE
activemq::commands::ActiveMQTopic, 553	activemq::commands::JournalTrace, 1769
ID_ BROKERID	ID_ JOURNALTRANSACTION
activemq::commands::BrokerId, 694	activemq::commands::JournalTransaction, 1793
ID_ BROKERINFO	ID_ KEEPALIVEINFO
activemq::commands::BrokerInfo, 721	activemq::commands::KeepAliveInfo, 1815
ID_ CONNECTIONCONTROL	ID_ LASTPARTIALCOMMAND
activemq::commands::ConnectionControl, 1059	activemq::commands::LastPartialCommand, 1842
ID_ CONNECTIONERROR	ID_ LOCALTRANSACTIONID
activemq::commands::ConnectionError, 1083	activemq::commands::LocalTransactionId, 1878
ID_ CONNECTIONID	ID_ MESSAGE
activemq::commands::ConnectionId, 1109	activemq::commands::Message, 2035
ID_ CONNECTIONINFO	ID_ MESSAGEACK
activemq::commands::ConnectionInfo, 1134	activemq::commands::MessageAck, 2060
ID_ CONSUMERCONTROL	ID_ MESSAGEDISPATCH
activemq::commands::ConsumerControl, 1170	activemq::commands::MessageDispatch, 2088
ID_ CONSUMERID	ID_ MESSAGEDISPATCHNOTIFICATION
activemq::commands::ConsumerId, 1194	activemq::commands::MessageDispatchNotification, 2120
ID_ CONSUMERINFO	ID_ MESSAGEID
activemq::commands::ConsumerInfo, 1222	activemq::commands::MessageId, 2146
ID_ CONTROLCOMMAND	ID_ MESSAGEPULL

- activemq::commands::MessagePull, 2207
- ID_NETWORKBRIDGEFILTER
 - activemq::commands::NetworkBridgeFilter, 2249
- ID_PARTIALCOMMAND
 - activemq::commands::PartialCommand, 2322
- ID_PRODUCERACK
 - activemq::commands::ProducerAck, 2423
- ID_PRODUCERID
 - activemq::commands::ProducerId, 2449
- ID_PRODUCERINFO
 - activemq::commands::ProducerInfo, 2474
- ID_REMOVEINFO
 - activemq::commands::RemoveInfo, 2540
- ID_REMOVESUBSCRIPTIONINFO
 - activemq::commands::RemoveSubscriptionInfo, 2564
- ID_REPLAYCOMMAND
 - activemq::commands::ReplayCommand, 2587
- ID_RESPONSE
 - activemq::commands::Response, 2614
- ID_SESSIONID
 - activemq::commands::SessionId, 2681
- ID_SESSIONINFO
 - activemq::commands::SessionInfo, 2705
- ID_SHUTDOWNINFO
 - activemq::commands::ShutdownInfo, 2759
- ID_SUBSCRIPTIONINFO
 - activemq::commands::SubscriptionInfo, 2911
- ID_TRANSACTIONID
 - activemq::commands::TransactionId, 3018
- ID_TRANSACTIONINFO
 - activemq::commands::TransactionInfo, 3040
- ID_WIREFORMATINFO
 - activemq::commands::WireFormatInfo, 3162
- ID_XATRANSACTIONID
 - activemq::commands::XATransactionId, 3197
- IllegalArgumentException
 - decaf::lang::exceptions::IllegalArgumentException, 1614, 1615
- IllegalMonitorStateException
 - decaf::lang::exceptions::IllegalMonitorStateException, 1616, 1617
- IllegalStateException
 - cms::IllegalStateException, 1621
 - decaf::lang::exceptions::IllegalStateException, 1619, 1620
- IllegalThreadStateException
 - decaf::lang::exceptions::IllegalThreadStateException, 1622, 1623
- InactivityMonitor
 - activemq::transport::inactivity::InactivityMonitor, 1625
- increaseUsage
 - activemq::util::MemoryUsage, 2016
 - activemq::util::Usage, 3138
- incrementAndGet
 - decaf::util::concurrent::atomic::AtomicInteger, 586
- indexOf
 - decaf::util::List, 1868
 - decaf::util::StlList, 2841
- IndexOutOfBoundsException
 - decaf::lang::exceptions::IndexOutOfBoundsException, 1628, 1629
- INDIVIDUAL_ACKNOWLEDGE
 - cms::Session, 2667
- Info
 - decaf::util::logging, 117
- info
 - decaf::util::logging::Logger, 1913
 - decaf::util::logging::SimpleLogger, 2785
- init
 - activemq::cmsutil::CmsAccessor, 957
 - activemq::cmsutil::CmsDestinationAccessor, 959
 - activemq::cmsutil::CmsTemplate, 975
 - activemq::cmsutil::DestinationResolver, 1416
 - activemq::cmsutil::DynamicDestinationResolver, 1472
- initCause
 - decaf::lang::Exception, 1481
 - decaf::lang::Throwable, 2986
- initializeLibrary
 - activemq::library::ActiveMQCPP, 239
- initializeRuntime
 - decaf::lang::Runtime, 2643, 2644
- inputStream
 - decaf::io::FilterInputStream, 1536
- inReceive
 - activemq::wireformat::openwire::OpenWireFormat, 2302
 - activemq::wireformat::stomp::StompWireFormat, 2885
 - activemq::wireformat::WireFormat, 3150
- insert
 - decaf::internal::util::TimerTaskHeap, 3004
- IntArrayBuffer
 - decaf::internal::nio::IntArrayBuffer, 1635, 1636
- intBitsToFloat

- decaf::lang::Float, 1549
- IntBuffer
 - decaf::nio::IntBuffer, 1644
- Integer
 - decaf::lang::Integer, 1655
- INTEGER_TYPE
 - activemq::util::PrimitiveValueNode, 2402
- IntegerResponse
 - activemq::commands::IntegerResponse, 1668
- IntegerResponseMarshaller
 - activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller, 1675
 - activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller, 1671
 - activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller, 1679
 - activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller, 1683
 - activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller, 1687
- InternalCommandListener
 - activemq::transport::mock::InternalCommandListener, 1690
- InterruptedException
 - decaf::lang::exceptions::InterruptedException, 1692
- InterruptedIOException
 - decaf::io::InterruptedIOException, 1694, 1695
- int Value
 - activemq::util::PrimitiveValueNode::PrimitiveValue, 2396
 - decaf::lang::Byte, 780
 - decaf::lang::Character, 918
 - decaf::lang::Double, 1446
 - decaf::lang::Float, 1549
 - decaf::lang::Integer, 1658
 - decaf::lang::Long, 1940
 - decaf::lang::Number, 2283
 - decaf::lang::Short, 2733
 - decaf::util::concurrent::atomic::AtomicInteger, 586
- InvalidClientIdException
 - cms::InvalidClientIdException, 1697
- InvalidDestinationException
 - cms::InvalidDestinationException, 1698
- InvalidKeyException
 - decaf::security::InvalidKeyException, 1699, 1700
- InvalidMarkException
 - decaf::nio::InvalidMarkException, 1701, 1702
- InvalidSelectorException
 - cms::InvalidSelectorException, 1704
- InvalidStateException
 - decaf::lang::exceptions::InvalidStateException, 1705, 1706
- IOException
 - decaf::io::IOException, 1707, 1708
- IOTransport
 - activemq::transport::IOTransport, 1711
- isAbsolute
 - decaf::internal::net::URIType, 3130
 - decaf::net::URI, 3103
- isAlwaysSyncSend
 - activemq::core::ActiveMQSession, 415
- isBrokerInfo
 - activemq::commands::BaseCommand, 599
 - activemq::commands::BrokerInfo, 718
 - activemq::commands::Command, 992
- isBrokerMasterConnector
 - activemq::commands::ConnectionInfo, 1132
- isCancelled
 - decaf::util::concurrent::Future, 1599
- isClientAcknowledge
 - activemq::core::ActiveMQSession, 415
- isClientMaster
 - activemq::commands::ConnectionInfo, 1132
- isClose
 - activemq::commands::ConnectionControl, 1058
 - activemq::commands::ConsumerControl, 1169
- isClosed
 - activemq::core::ActiveMQConnection, 208
- isConsumer
 - activemq::commands::ConsumerInfo, 1219
- isBusy
 - decaf::util::concurrent::PooledThread, 2365
- isCacheEnabled
 - activemq::commands::WireFormatInfo, 3158
 - activemq::wireformat::openwire::OpenWireFormat, 2302

- activemq::core::ActiveMQConsumer, 235
- activemq::core::ActiveMQProducer, 371
- activemq::core::MessageDispatchChannel, 2091
- activemq::transport::failover::BackupTransportPool, 593
- activemq::transport::failover::FailoverTransport, 1517
- activemq::transport::IOTransport, 1712
- activemq::transport::mock::MockTransport, 2232
- activemq::transport::tcp::TcpTransport, 2968
- activemq::transport::Transport, 3068
- activemq::transport::TransportFilter, 3076
- decaf::io::FilterInputStream, 1530
- decaf::io::FilterOutputStream, 1539
- isComposite
 - activemq::commands::ActiveMQDestination, 245
- isCompressed
 - activemq::commands::Message, 2029
- isConnected
 - activemq::transport::failover::FailoverTransport, 1517
 - activemq::transport::IOTransport, 1712
 - activemq::transport::mock::MockTransport, 2232
 - activemq::transport::tcp::TcpTransport, 2968
 - activemq::transport::Transport, 3068
 - activemq::transport::TransportFilter, 3076
 - decaf::net::BufferedSocket, 759
 - decaf::net::Socket, 2790
 - decaf::net::TcpSocket, 2964
- isConnectionAdvisory
 - activemq::commands::ActiveMQDestination, 245
- isConnectionInfo
 - activemq::commands::BaseCommand, 600
 - activemq::commands::Command, 992
 - activemq::commands::ConnectionInfo, 1132
- isConsumerAdvisory
 - activemq::commands::ActiveMQDestination, 246
- isConsumerInfo
 - activemq::commands::BaseCommand, 600
 - activemq::commands::Command, 992
 - activemq::commands::ConsumerInfo, 1219
- isDeletedByBroker
 - activemq::commands::ActiveMQBlobMessage, 145
- isDigit
 - decaf::lang::Character, 918
- isDispatchAsync
 - activemq::commands::ConsumerInfo, 1219
 - activemq::commands::ProducerInfo, 2472
- isDone
 - decaf::util::concurrent::Future, 1600
- isDroppable
 - activemq::commands::Message, 2029
- isDuplexConnection
 - activemq::commands::BrokerInfo, 718
- isDupsOkAcknowledge
 - activemq::core::ActiveMQSession, 415
- isEmpty
 - activemq::core::ActiveMQSessionExecutor, 421
 - activemq::core::MessageDispatchChannel, 2091
 - activemq::util::ActiveMQProperties, 377
 - cms::CMSProperties, 967
 - decaf::internal::util::TimerTaskHeap, 3004
 - decaf::util::AbstractCollection, 125
 - decaf::util::Collection, 987
 - decaf::util::concurrent::ConcurrentStlMap, 1032
 - decaf::util::concurrent::SynchronousQueue, 2950
 - decaf::util::Map, 1976
 - decaf::util::Properties, 2498
 - decaf::util::StlList, 2841
 - decaf::util::StlMap, 2852
 - decaf::util::StlSet, 2870
- isEnabled
 - activemq::transport::failover::BackupTransportPool, 595
- isExclusive
 - activemq::commands::ActiveMQDestination, 246
 - activemq::commands::ConsumerInfo, 1220
- isExit
 - activemq::commands::ConnectionControl, 1058
- isExpired
 - activemq::commands::Message, 2029
- isExplicitQosEnabled
 - activemq::cmsutil::CmsTemplate, 975
- isFailOnClose
 - activemq::transport::mock::MockTransport, 2232
- isFailOnKeepAliveSends
 - activemq::transport::mock::MockTransport, 2232
- isFailOnReceiveMessage
 - activemq::transport::mock::MockTransport, 2232

- isFailOnSendMessage
 - activemq::transport::mock::MockTransport, 2232
- isFailOnStart
 - activemq::transport::mock::MockTransport, 2232
- isFailOnStop
 - activemq::transport::mock::MockTransport, 2232
- isFair
 - decaf::util::concurrent::locks::ReentrantLock, 2528
 - decaf::util::concurrent::Semaphore, 2656
- isFaultTolerant
 - activemq::commands::ConnectionControl, 1058
 - activemq::transport::failover::FailoverTransport, 1517
 - activemq::transport::IOTransport, 1712
 - activemq::transport::mock::MockTransport, 2232
 - activemq::transport::tcp::TcpTransport, 2969
 - activemq::transport::Transport, 3068
 - activemq::transport::TransportFilter, 3076
- isFaultTolerantConfiguration
 - activemq::commands::BrokerInfo, 719
- isFlush
 - activemq::commands::ConsumerControl, 1169
- isFull
 - activemq::util::MemoryUsage, 2017
 - activemq::util::Usage, 3138
- isHeldByCurrentThread
 - decaf::util::concurrent::locks::ReentrantLock, 2529
- isIndividualAcknowledge
 - activemq::core::ActiveMQSession, 415
- isInfinite
 - decaf::lang::Double, 1447
 - decaf::lang::Float, 1550
- isInitialized
 - activemq::transport::failover::FailoverTransport, 1517
- isInTransaction
 - activemq::core::ActiveMQTransactionContext, 575
- isISOControl
 - decaf::lang::Character, 919
- isKeepAliveInfo
 - activemq::commands::BaseCommand, 600
 - activemq::commands::Command, 992
 - activemq::commands::KeepAliveInfo, 1814
- isKeepAliveResponseRequired
 - activemq::transport::inactivity::InactivityMonitor, 1626
- isLetter
 - decaf::lang::Character, 919
- isLetterOrDigit
 - decaf::lang::Character, 919
- isLocked
 - decaf::util::concurrent::Lock, 1904
 - decaf::util::concurrent::locks::ReentrantLock, 2529
- isLoggable
 - decaf::util::logging::Filter, 1527
 - decaf::util::logging::Handler, 1606
 - decaf::util::logging::Logger, 1913
 - decaf::util::logging::StreamHandler, 2891
- isLowerCase
 - decaf::lang::Character, 919
- isManageable
 - activemq::commands::ConnectionInfo, 1132
- isMarshalAware
 - activemq::commands::ActiveMQMapMessage, 280
 - activemq::commands::BaseDataStructure, 661
 - activemq::commands::Message, 2029
 - activemq::commands::WireFormatInfo, 3158
 - activemq::wireformat::MarshalAware, 1994
- isMasterBroker
 - activemq::commands::BrokerInfo, 719
- isMessage
 - activemq::commands::BaseCommand, 600
 - activemq::commands::Command, 992
 - activemq::commands::Message, 2030
- isMessageAck
 - activemq::commands::BaseCommand, 600
 - activemq::commands::Command, 993
 - activemq::commands::MessageAck, 2058
- isMessageDispatch
 - activemq::commands::BaseCommand, 600
 - activemq::commands::Command, 993
 - activemq::commands::MessageDispatch, 2087
- isMessageDispatchNotification
 - activemq::commands::BaseCommand, 600
 - activemq::commands::Command, 993
 - activemq::commands::MessageDispatchNotification, 2118
- isMessageIdEnabled
 - activemq::cmsutil::CmsTemplate, 976
- isMessageTimestampEnabled
 - activemq::cmsutil::CmsTemplate, 976
- isNaN

- decaf::lang::Double, 1447
- decaf::lang::Float, 1550
- isNetworkConnection
 - activemq::commands::BrokerInfo, 719
- isNetworkSubscription
 - activemq::commands::ConsumerInfo, 1220
- isNoLocal
 - activemq::cmsutil::CmsTemplate, 976
 - activemq::commands::ConsumerInfo, 1220
- isNoRangeAcks
 - activemq::commands::ConsumerInfo, 1220
- isOpaque
 - decaf::internal::net::URIType, 3130
 - decaf::net::URI, 3104
- isOptimizedAcknowledge
 - activemq::commands::ConsumerInfo, 1220
- isOrdered
 - activemq::commands::ActiveMQDestination, 246
- isPending
 - activemq::threads::CompositeTask, 1016
 - activemq::transport::failover::BackupTransportPool, 595
 - activemq::transport::failover::CloseTransportsTask, 953
 - activemq::transport::failover::FailoverTransport, 1518
- isPersistent
 - activemq::commands::Message, 2030
- isPrepared
 - activemq::state::TransactionState, 3061
- isProducerAck
 - activemq::commands::BaseCommand, 601
 - activemq::commands::Command, 993
 - activemq::commands::ProducerAck, 2422
- isProducerAdvisory
 - activemq::commands::ActiveMQDestination, 246
- isProducerInfo
 - activemq::commands::BaseCommand, 601
 - activemq::commands::Command, 993
 - activemq::commands::ProducerInfo, 2472
- isPubSubDomain
 - activemq::cmsutil::CmsDestinationAccessor, 959
- isQueue
 - activemq::commands::ActiveMQDestination, 246
- isRandomize
 - activemq::transport::failover::FailoverTransport, 1518
 - activemq::transport::failover::URIPool, 3119
- isReadOnly
 - decaf::internal::nio::ByteBuffer, 821
 - decaf::internal::nio::CharArrayBuffer, 928
 - decaf::internal::nio::DoubleArrayBuffer, 1458
 - decaf::internal::nio::FloatArrayBuffer, 1560
 - decaf::internal::nio::IntArrayBuffer, 1639
 - decaf::internal::nio::LongArrayBuffer, 1953
 - decaf::internal::nio::ShortArrayBuffer, 2744
 - decaf::nio::Buffer, 745
 - decaf::nio::ByteBuffer, 864
- isReadOnlyBody
 - activemq::commands::Message, 2030
- isReadOnlyProperties
 - activemq::commands::Message, 2030
- isRecievedByDFBridge
 - activemq::commands::Message, 2030
- isRemoveInfo
 - activemq::commands::BaseCommand, 601
 - activemq::commands::Command, 993
 - activemq::commands::RemoveInfo, 2539
- isRemoveSubscriptionInfo
 - activemq::commands::BaseCommand, 601
 - activemq::commands::Command, 993
 - activemq::commands::RemoveSubscriptionInfo, 2563
- isResponse
 - activemq::commands::BaseCommand, 601
 - activemq::commands::Command, 993
 - activemq::commands::Response, 2613
- isResponseRequired
 - activemq::commands::BaseCommand, 601
 - activemq::commands::Command, 994
- isRestoreConsumers
 - activemq::state::ConnectionStateTracker, 1162
- isRestoreProducers
 - activemq::state::ConnectionStateTracker, 1162
- isRestoreSessions
 - activemq::state::ConnectionStateTracker, 1162
- isRestoreTransaction
 - activemq::state::ConnectionStateTracker, 1162
- isResume
 - activemq::commands::ConnectionControl, 1058
- isRetroactive
 - activemq::commands::ConsumerInfo, 1220
- isRunning
 - activemq::core::ActiveMQSessionExecutor, 421

- activemq::core::MessageDispatchChannel, 2091
- isScheduled
 - decaf::util::TimerTask, 3002
- isServerAuthority
 - decaf::internal::net::URIType, 3130
- isShutdownInfo
 - activemq::commands::BaseCommand, 602
 - activemq::commands::Command, 994
 - activemq::commands::ShutdownInfo, 2759
- isSizePrefixDisabled
 - activemq::commands::WireFormatInfo, 3158
 - activemq::wireformat::openwire::OpenWireFormat, 2302
- isSlaveBroker
 - activemq::commands::BrokerInfo, 719
- isStackTraceEnabled
 - activemq::commands::WireFormatInfo, 3159
 - activemq::wireformat::openwire::OpenWireFormat, 2302
- isStart
 - activemq::commands::ConsumerControl, 1169
- isStarted
 - activemq::core::ActiveMQConnection, 208
 - activemq::core::ActiveMQSession, 415
- isStop
 - activemq::commands::ConsumerControl, 1169
- isSuspend
 - activemq::commands::ConnectionControl, 1058
- isSynchronizationRegistered
 - activemq::core::ActiveMQConsumer, 236
- isTcpNoDelayEnabled
 - activemq::commands::WireFormatInfo, 3159
 - activemq::wireformat::openwire::OpenWireFormat, 2303
- isTemporary
 - activemq::commands::ActiveMQDestination, 246
- isTightEncodingEnabled
 - activemq::commands::WireFormatInfo, 3159
 - activemq::wireformat::openwire::OpenWireFormat, 2303
- isTopic
 - activemq::commands::ActiveMQDestination, 247
- isTrackMessages
 - activemq::state::ConnectionStateTracker, 1162
 - activemq::transport::failover::FailoverTransport, 1518
- isTrackTransactions
 - activemq::state::ConnectionStateTracker, 1162
- isTransacted
 - activemq::cmsutil::PooledSession, 2362
 - activemq::core::ActiveMQSession, 415
 - cms::Session, 2674
- isTransactionInfo
 - activemq::commands::BaseCommand, 602
 - activemq::commands::Command, 994
 - activemq::commands::TransactionInfo, 3039
- isUpperCase
 - decaf::lang::Character, 919
- isUseAsyncSend
 - activemq::core::ActiveMQConnectionSupport, 224
- isUseExponentialBackOff
 - activemq::transport::failover::FailoverTransport, 1518
- isValid
 - activemq::commands::WireFormatInfo, 3159
 - decaf::internal::net::URIType, 3130
- isValidDomainName
 - decaf::internal::net::URIHelper, 3112
- isValidHexChar
 - decaf::internal::net::URIHelper, 3112
- isValidHost
 - decaf::internal::net::URIHelper, 3112
- isValidIP4Word
 - decaf::internal::net::URIHelper, 3113
- isValidIP6Address
 - decaf::internal::net::URIHelper, 3113
- isValidIPv4Address
 - decaf::internal::net::URIHelper, 3113
- isWaitingForResponse
 - activemq::state::Tracked, 3015
- isWhitespace
 - decaf::lang::Character, 919
- isWildcard
 - activemq::commands::ActiveMQDestination, 247
 - activemq::commands::WireFormatInfo, 3159
 - activemq::commands::BaseCommand, 602
 - activemq::commands::Command, 994
 - activemq::commands::WireFormatInfo, 3159
- itemExists

- activemq::commands::ActiveMQMapMessage, 280
- cms::MapMessage, 1988
- iterate
 - activemq::core::ActiveMQConsumer, 236
 - activemq::core::ActiveMQSessionExecutor, 421
 - activemq::threads::CompositeTaskRunner, 1018
 - activemq::threads::Task, 2956
 - activemq::transport::failover::BackupTransportPool, 596
 - activemq::transport::failover::CloseTransportsTask, 953
 - activemq::transport::failover::FailoverTransport, 1518
- iterator
 - decaf::lang::Iterable, 1716
 - decaf::util::concurrent::SynchronousQueue, 2950
 - decaf::util::PriorityQueue, 2417
 - decaf::util::StlList, 2842
 - decaf::util::StlQueue, 2861
 - decaf::util::StlSet, 2870
- JournalQueueAck
 - activemq::commands::JournalQueueAck, 1719
- JournalQueueAckMarshaller
 - activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller, 1735
 - activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller, 1723
 - activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller, 1731
 - activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller, 1727
 - activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller, 1739
- JournalTopicAck
 - activemq::commands::JournalTopicAck, 1743
- JournalTopicAckMarshaller
 - activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller, 1752
 - activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller, 1748
 - activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller, 1756
 - activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller, 1760
 - activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller, 1764
- JournalTrace
 - activemq::commands::JournalTrace, 1767
 - JournalTraceMarshaller
 - activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller, 1783
 - activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller, 1771
 - activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller, 1779
 - activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller, 1775
 - activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller, 1787
 - JournalTransaction
 - activemq::commands::JournalTransaction, 1791
 - JournalTransactionMarshaller
 - activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller, 1806
 - activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller, 1794
 - activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller, 1798
 - activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller, 1802
 - activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller, 1810
 - KeepAliveInfo
 - activemq::commands::KeepAliveInfo, 1813
 - KeepAliveInfoMarshaller
 - activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller, 1836
 - activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller, 1817
 - activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller, 1826
 - activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller, 1820
 - activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller, 1821
 - KeyException
 - decaf::security::KeyException, 1838, 1839
 - keySet
 - decaf::util::StlMap, 1032
 - lastDeliveredSequenceId
 - activemq::core::ActiveMQConsumer, 1977
 - lastIndexOf
 - decaf::util::StlList, 2842
 - lastMessageId
 - activemq::core::ActiveMQConsumer, 1977

- activemq::commands::MessageAck, 2060
- lastNakNumber
 - activemq::commands::ReplayCommand, 2587
- LastPartialCommand
 - activemq::commands::LastPartialCommand, 1841
- LastPartialCommandMarshaller
 - activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller, 1848
 - activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller, 1844
 - activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller, 1852
 - activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller, 1856
 - activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller, 1860
- length
 - decaf::lang::CharSequence, 947
 - decaf::nio::CharBuffer, 940
- Less
 - decaf::util::comparators::Less, 1863
- Level
 - decaf::util::logging, 117
- limit
 - decaf::nio::Buffer, 745
- LineNumber
 - activemq::commands::BrokerError::StackTraceElement, 2812
- LIST_TYPE
 - activemq::util::PrimitiveValueNode, 2402
- listener
 - activemq::transport::TransportFilter, 3080
- listIterator
 - decaf::util::List, 1869, 1870
 - decaf::util::StlList, 2842, 2843
- listValue
 - activemq::util::PrimitiveValueNode::PrimitiveValueNode, 2396
- load
 - decaf::util::Properties, 2498, 2500
- LocalTransactionId
 - activemq::commands::LocalTransactionId, 1876
- LocalTransactionIdMarshaller
 - activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller, 1891
 - activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller, 1879
 - activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller, 1883
 - activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller, 1887
- activemq::wireformat::openwire::marshal::v5::LocalTransactionIdMarshaller, 1895
- Lock
 - decaf::util::concurrent::Lock, 1904
- lock
 - activemq::core::MessageDispatchChannel, 2092
 - decaf::internal::io::StandardErrorOutputStream, 2804
 - decaf::internal::io::StandardInputStream, 2820
 - decaf::internal::io::StandardOutputStream, 2827
 - decaf::internal::util::concurrent::MutexImpl, 2205
 - decaf::internal::util::concurrent::SynchronizableImpl, 2205
 - decaf::io::BlockingByteArrayInputStream, 668
 - decaf::io::ByteArrayInputStream, 830
 - decaf::io::ByteArrayOutputStream, 838
 - decaf::io::FilterInputStream, 1531
 - decaf::io::FilterOutputStream, 1539
 - decaf::net::SocketInputStream, 2798
 - decaf::net::SocketOutputStream, 2805
 - decaf::util::AbstractCollection, 126
 - decaf::util::concurrent::ConcurrentStlMap, 1033
 - decaf::util::concurrent::Lock, 1904
 - decaf::util::concurrent::locks::Lock, 1900
 - decaf::util::concurrent::locks::ReentrantLock, 2529
 - decaf::util::concurrent::Mutex, 2240
 - decaf::util::concurrent::Synchronizable, 2932
 - decaf::util::StlMap, 2853
 - decaf::util::StlQueue, 2862
- lock_count
 - decaf::util::concurrent::MutexHandle, 2244
- lock_owner
 - decaf::util::concurrent::MutexHandle, 2244
- lockInterruptibly
 - decaf::util::concurrent::locks::Lock, 1900
 - decaf::util::concurrent::locks::ReentrantLock, 2530
- log
 - decaf::util::logging::Log, 1914, 1915
 - decaf::util::logging::LogWriter, 1933
 - decaf::util::logging::SimpleLogger, 2785
- LOGDECAF_DEBUG
 - decaf::util::logging::Log, 1914, 1915
- LOGDECAF_DEBUG_1
 - decaf::util::logging::Log, 1914, 1915
- LOGDECAF_DECLARE
 - decaf::util::logging::Log, 1914, 1915

- LoggerDefines.h, 3639
- LOGDECAF_DECLARE_LOCAL
 - LoggerDefines.h, 3640
- LOGDECAF_ERROR
 - LoggerDefines.h, 3640
- LOGDECAF_FATAL
 - LoggerDefines.h, 3640
- LOGDECAF_INFO
 - LoggerDefines.h, 3640
- LOGDECAF_INITIALIZE
 - LoggerDefines.h, 3640
- LOGDECAF_WARN
 - LoggerDefines.h, 3640
- Logger
 - decaf::util::logging::Logger, 1910
- LoggerDefines.h
 - LOGDECAF_DEBUG, 3639
 - LOGDECAF_DEBUG_1, 3639
 - LOGDECAF_DECLARE, 3639
 - LOGDECAF_DECLARE_LOCAL, 3640
 - LOGDECAF_ERROR, 3640
 - LOGDECAF_FATAL, 3640
 - LOGDECAF_INFO, 3640
 - LOGDECAF_INITIALIZE, 3640
 - LOGDECAF_WARN, 3640
- LoggerHierarchy
 - decaf::util::logging::LoggerHierarchy, 1917
- LoggingInputStream
 - activemq::io::LoggingInputStream, 1917
- LoggingOutputStream
 - activemq::io::LoggingOutputStream, 1919
- LoggingTransport
 - activemq::transport::logging::LoggingTransport, 1921
- LogManager
 - decaf::util::logging::LogManager, 1925
- LogRecord
 - decaf::util::logging::LogRecord, 1929
- LogWriter
 - decaf::util::logging::LogWriter, 1933
- Long
 - decaf::lang::Long, 1937
- LONG_TYPE
 - activemq::util::PrimitiveValueNode, 2402
- LongArrayBuffer
 - decaf::internal::nio::LongArrayBuffer, 1949, 1950
- longBitsToDouble
 - decaf::lang::Double, 1447
- LongBuffer
 - decaf::nio::LongBuffer, 1958
- LongSequenceGenerator
 - activemq::util::LongSequenceGenerator, 1967
- longValue
 - activemq::util::PrimitiveValueNode::PrimitiveValue, 2396
 - decaf::lang::Byte, 780
 - decaf::lang::Character, 919
 - decaf::lang::Double, 1448
 - decaf::lang::Float, 1550
 - decaf::lang::Integer, 1658
 - decaf::lang::Long, 1940
 - decaf::lang::Number, 2283
 - decaf::lang::Short, 2734
 - decaf::util::concurrent::atomic::AtomicInteger, 586
- lookup
 - decaf::security_-
provider::SecurityProviderMap, 2649
- looseMarshal
 - activemq::wireformat::openwire::marshal::BaseDataStreamMa
643
 - activemq::wireformat::openwire::marshal::DataStreamMarshal
1342
 - activemq::wireformat::openwire::marshal::v1::ActiveMQBlobM
152
 - activemq::wireformat::openwire::marshal::v1::ActiveMQBytesI
187
 - activemq::wireformat::openwire::marshal::v1::ActiveMQDestin
255
 - activemq::wireformat::openwire::marshal::v1::ActiveMQMapM
290
 - activemq::wireformat::openwire::marshal::v1::ActiveMQMessa
312
 - activemq::wireformat::openwire::marshal::v1::ActiveMQObject
352
 - activemq::wireformat::openwire::marshal::v1::ActiveMQQueue
388
 - activemq::wireformat::openwire::marshal::v1::ActiveMQStream
441
 - activemq::wireformat::openwire::marshal::v1::ActiveMQTempI
463
 - activemq::wireformat::openwire::marshal::v1::ActiveMQTempO
486
 - activemq::wireformat::openwire::marshal::v1::ActiveMQTempS
510
 - activemq::wireformat::openwire::marshal::v1::ActiveMQTextM
535
 - activemq::wireformat::openwire::marshal::v1::ActiveMQTopicM
559
 - activemq::wireformat::openwire::marshal::v1::BaseCommandM
611
 - activemq::wireformat::openwire::marshal::v1::BrokerIdMarshal
699
 - activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshal
727

activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller; openwire::marshal::v1::MessagePullMarshaller 1065
 activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller; openwire::marshal::v1::NetworkBridgeFailureMarshaller 1089
 activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller; openwire::marshal::v1::PartialCommandMarshaller 1114
 activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller; openwire::marshal::v1::ProducerAckMarshaller 1144
 activemq::wireformat::openwire::marshal::v1::ConsumerGroupViewMarshaller; openwire::marshal::v1::ProducerIdMarshaller 1180
 activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller; openwire::marshal::v1::ProducerInfoMarshaller 1204
 activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller; openwire::marshal::v1::RemoveInfoMarshaller 1228
 activemq::wireformat::openwire::marshal::v1::ContainerCommandMarshaller; openwire::marshal::v1::RemoveSubscriptionMarshaller 1256
 activemq::wireformat::openwire::marshal::v1::DataActiveResponseMarshaller; openwire::marshal::v1::ReplayCommandMarshaller 1280
 activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller; openwire::marshal::v1::ResponseMarshaller 1315
 activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller; openwire::marshal::v1::SessionIdMarshaller 1408
 activemq::wireformat::openwire::marshal::v1::DisconnectViewMarshaller; openwire::marshal::v1::SessionInfoMarshaller 1437
 activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller; openwire::marshal::v1::ShutdownInfoMarshaller 1492
 activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller; openwire::marshal::v1::SubscriptionInfoMarshaller 1581
 activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller; openwire::marshal::v1::TransactionIdMarshaller 1675
 activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller; openwire::marshal::v1::TransactionInfoMarshaller 1735
 activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller; openwire::marshal::v1::WireFormatInfoMarshaller 1752
 activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller; openwire::marshal::v1::XATransactionInfoMarshaller 1783
 activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller; openwire::marshal::v2::ActiveMQBlobMarshaller 1806
 activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller; openwire::marshal::v2::ActiveMQBytesMarshaller 1833
 activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller; openwire::marshal::v2::ActiveMQDestinationMarshaller 1848
 activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller; openwire::marshal::v2::ActiveMQMapMarshaller 1891
 activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller; openwire::marshal::v2::ActiveMQMessageMarshaller 2077
 activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller; openwire::marshal::v2::ActiveMQObjectMarshaller 2109
 activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller; openwire::marshal::v2::ActiveMQQueueMarshaller 2133
 activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller; openwire::marshal::v2::ActiveMQStreamMarshaller 2163
 activemq::wireformat::openwire::marshal::v1::MessageMarshaller; openwire::marshal::v2::ActiveMQTempMarshaller 2184

activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller; openwire::marshal::v3::FlushCommand	251	1585
activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller; openwire::marshal::v3::IntegerResponse	286	1679
activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller; openwire::marshal::v3::JournalQueueAck	309	1731
activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller; openwire::marshal::v3::JournalTopicAck	348	1756
activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller; openwire::marshal::v3::JournalTraceMa	384	1779
activemq::wireformat::openwire::marshal::v3::ActiveMQSequenceMessageMarshaller; openwire::marshal::v3::JournalTransact	437	1798
activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller; openwire::marshal::v3::KeepAliveInfoM	460	1825
activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller; openwire::marshal::v3::LastPartialComm	483	1852
activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller; openwire::marshal::v3::LocalTransaction	506	1883
activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller; openwire::marshal::v3::MessageAckMar	531	2069
activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller; openwire::marshal::v3::MessageDispatch	555	2105
activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller; openwire::marshal::v3::MessageDispatch	604	2129
activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller; openwire::marshal::v3::MessageIdMarsh	695	2155
activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller; openwire::marshal::v3::MessageMarsh	723	2176
activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller; openwire::marshal::v3::MessagePullMar	1061	2212
activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller; openwire::marshal::v3::NetworkBridgeF	1085	2255
activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller; openwire::marshal::v3::PartialCommand	1110	2328
activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller; openwire::marshal::v3::ProducerAckMar	1136	2429
activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller; openwire::marshal::v3::ProducerIdMars	1172	2455
activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller; openwire::marshal::v3::ProducerInfoMa	1196	2479
activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller; openwire::marshal::v3::RemoveInfoMars	1224	2553
activemq::wireformat::openwire::marshal::v3::ContentCommandMarshaller; openwire::marshal::v3::RemoveSubscrip	1248	2577
activemq::wireformat::openwire::marshal::v3::DataActiveResponseMarshaller; openwire::marshal::v3::ReplayCommand	1272	2593
activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller; openwire::marshal::v3::ResponseMarsha	1311	2626
activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller; openwire::marshal::v3::SessionIdMarsha	1401	2699
activemq::wireformat::openwire::marshal::v3::DiscardRequestMarshaller; openwire::marshal::v3::SessionInfoMars	1425	2722
activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller; openwire::marshal::v3::ShutdownInfoM	1496	2769

activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller	2917	1232	openwire::marshal::v4::ConsumerInfoMarshaller
activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller	3034	1252	openwire::marshal::v4::ControlCommandMarshaller
activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller	3046	1276	openwire::marshal::v4::DataArrayResponseMarshaller
activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller	3180	1323	openwire::marshal::v4::DataResponseMarshaller
activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller	3202	1404	openwire::marshal::v4::DestinationInfoMarshaller
activemq::wireformat::openwire::marshal::v4::ActiveMQBlobWireFormatMarshaller	156	1429	openwire::marshal::v4::DiscoveryEventMarshaller
activemq::wireformat::openwire::marshal::v4::ActiveMQByteWireFormatMarshaller	191	1500	openwire::marshal::v4::ExceptionResponseMarshaller
activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller	259	1589	openwire::marshal::v4::FlushCommandMarshaller
activemq::wireformat::openwire::marshal::v4::ActiveMQMapWireFormatMarshaller	294	1683	openwire::marshal::v4::IntegerResponseMarshaller
activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller	316	1727	openwire::marshal::v4::JournalQueueAckMarshaller
activemq::wireformat::openwire::marshal::v4::ActiveMQObjectWireFormatMarshaller	356	1760	openwire::marshal::v4::JournalTopicAckMarshaller
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueWireFormatMarshaller	392	1775	openwire::marshal::v4::JournalTraceMarshaller
activemq::wireformat::openwire::marshal::v4::ActiveMQSequenceWireFormatMarshaller	445	1802	openwire::marshal::v4::JournalTransactionMarshaller
activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller	467	1829	openwire::marshal::v4::KeepAliveInfoMarshaller
activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller	490	1856	openwire::marshal::v4::LastPartialCommandMarshaller
activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller	514	1887	openwire::marshal::v4::LocalTransactionMarshaller
activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller	539	2073	openwire::marshal::v4::MessageAckMarshaller
activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller	563	2101	openwire::marshal::v4::MessageDispatchMarshaller
activemq::wireformat::openwire::marshal::v4::BaseCommandWireFormatMarshaller	618	2126	openwire::marshal::v4::MessageDispatchMarshaller
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller	703	2151	openwire::marshal::v4::MessageIdMarshaller
activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller	731	2172	openwire::marshal::v4::MessageMarshaller
activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller	1069	2224	openwire::marshal::v4::MessagePullMarshaller
activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller	1093	2267	openwire::marshal::v4::NetworkBridgeFailureMarshaller
activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller	1118	2332	openwire::marshal::v4::PartialCommandMarshaller
activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller	1140	2433	openwire::marshal::v4::ProducerAckMarshaller
activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller	1176	2467	openwire::marshal::v4::ProducerIdMarshaller
activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller	1200	2487	openwire::marshal::v4::ProducerInfoMarshaller

activemq::wireformat::openwire::marshal::v4::RemoteInfoMarshaller	735	activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller	735
2557		activemq::wireformat::openwire::marshal::v4::RemoteSubscriptionInfoMarshaller	1073
2581		activemq::wireformat::openwire::marshal::v4::ReplyCommandMarshaller	1096
2600		activemq::wireformat::openwire::marshal::v4::ResponseMarshaller	1122
2639		activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller	1148
2695		activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller	1184
2714		activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller	1208
2765		activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller	1236
2924		activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller	1259
3027		activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller	1284
3054		activemq::wireformat::openwire::marshal::v4::WireFormatMarshaller	1319
3176		activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller	1412
3206		activemq::wireformat::openwire::marshal::v5::ActiveMQBlockWireFormatMarshaller	1433
160		activemq::wireformat::openwire::marshal::v5::ActiveMQByteMessageMarshaller	1504
195		activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller	1593
263		activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller	1687
298		activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller	1739
320		activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller	1764
360		activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller	1787
396		activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller	1810
449		activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller	1821
471		activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller	1860
494		activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller	1895
518		activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller	2065
543		activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller	2113
566		activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller	2137
624		activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller	2159
707			

activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller,	644
2180	
activemq::wireformat::openwire::marshal::v5::MessageIdStringMarshaller,	645
2220	
activemq::wireformat::openwire::marshal::v5::NetworkFilterMarshaller,	645
2259	
looseUnmarshal	
activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller,	645
2340	
activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller,	645
2437	
activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller,	1348
2459	
activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller,	152
2483	
activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller,	188
2545	
activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller,	255
2545	
activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller,	290
2566	
activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller,	313
2596	
activemq::wireformat::openwire::marshal::v5::ResponseMarshaller,	353
2630	
activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller,	388
2683	
activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller,	441
2718	
activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller,	464
2777	
activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller,	487
2920	
activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller,	511
3031	
activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller,	535
3042	
activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller,	559
3168	
activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller,	612
3214	
looseMarshalBrokerError	700
activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller,	727
643	
looseMarshalCachedObject	1065
activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller,	1065
643	
looseMarshalLong	1089
activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller,	1115
644	
looseMarshalNestedObject	1111
activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller,	1111
644	
activemq::wireformat::openwire::OpenWireFormat,	1180
2303	
looseMarshalObjectArray	1204

activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller 2542
 1228
 activemq::wireformat::openwire::marshal::v1::ContactCommandMarshaller 2574
 1256
 activemq::wireformat::openwire::marshal::v1::DataActiveResponseMarshaller 2605
 1281
 activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller 2635
 1316
 activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller 2687
 1409
 activemq::wireformat::openwire::marshal::v1::DiscoveryRequestMarshaller 2711
 1437
 activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller 2761
 1493
 activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller 2913
 1581
 activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller 3024
 1676
 activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller 3050
 1735
 activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller 3172
 1752
 activemq::wireformat::openwire::marshal::v1::JournalTransactionInfoMarshaller 3211
 1783
 activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller 164
 1806
 activemq::wireformat::openwire::marshal::v1::KeepActiveInfoMarshaller 200
 1833
 activemq::wireformat::openwire::marshal::v1::LastReceivedCommandMarshaller 267
 1849
 activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller 302
 1892
 activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller 325
 2078
 activemq::wireformat::openwire::marshal::v1::MessageDispatchInfoMarshaller 365
 2109
 activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller 400
 2134
 activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller 453
 2163
 activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller 475
 2184
 activemq::wireformat::openwire::marshal::v1::MessagePullInfoMarshaller 499
 2217
 activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller 523
 2263
 activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller 547
 2337
 activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller 571
 2441
 activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller 632
 2463
 activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller 711
 2492

activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller	2168
739	
activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller	2209
1077	
activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller	2252
1101	
activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller	2324
1126	
activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller	2425
1152	
activemq::wireformat::openwire::marshal::v2::ConsumerGroupViewMarshaller	2452
1188	
activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller	2476
1212	
activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller	2550
1240	
activemq::wireformat::openwire::marshal::v2::ContainerCommandMarshaller	2570
1264	
activemq::wireformat::openwire::marshal::v2::DataAndResponseMarshaller	2589
1289	
activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller	2622
1328	
activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller	2691
1397	
activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller	2707
1422	
activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller	2773
1489	
activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller	2928
1578	
activemq::wireformat::openwire::marshal::v2::IntegrationResponseMarshaller	3020
1672	
activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller	3058
1724	
activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller	3165
1748	
activemq::wireformat::openwire::marshal::v2::JournalTransactionIdMarshaller	3199
1771	
activemq::wireformat::openwire::marshal::v2::JournalTransactionIdMarshaller	148
1795	
activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller	184
1817	
activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller	252
1845	
activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller	286
1880	
activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller	309
2062	
activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller	349
2097	
activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller	384
2122	
activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller	438
2148	

activemq::wireformat::openwire::marshal::v3::ActiveMQTempDeflationMarshaller	460	activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller	1825
activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller	483	activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller	1853
activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller	507	activemq::wireformat::openwire::marshal::v3::LocalTransactionMarshaller	1884
activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller	531	activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller	2070
activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller	555	activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller	2105
activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller	605	activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller	2130
activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller	696	activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller	2156
activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller	723	activemq::wireformat::openwire::marshal::v3::MessageMarshaller	2176
activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller	1061	activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller	2213
activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller	1085	activemq::wireformat::openwire::marshal::v3::NetworkBridgeFailureMarshaller	2255
activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller	1111	activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller	2328
activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller	1136	activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller	2429
activemq::wireformat::openwire::marshal::v3::ConsumerGroupMarshaller	1172	activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller	2456
activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller	1197	activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller	2480
activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller	1225	activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller	2554
activemq::wireformat::openwire::marshal::v3::ContactCommandMarshaller	1248	activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionMarshaller	2578
activemq::wireformat::openwire::marshal::v3::DataActiveResponseMarshaller	1273	activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller	2593
activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller	1312	activemq::wireformat::openwire::marshal::v3::ResponseMarshaller	2626
activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller	1401	activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller	2699
activemq::wireformat::openwire::marshal::v3::DiscardResponseMarshaller	1426	activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller	2723
activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller	1497	activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller	2769
activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller	1585	activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller	2917
activemq::wireformat::openwire::marshal::v3::IntegratedResponseMarshaller	1680	activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller	3035
activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller	1731	activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller	3046
activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller	1756	activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller	3180
activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller	1779	activemq::wireformat::openwire::marshal::v3::XATransactionInfoMarshaller	3203
activemq::wireformat::openwire::marshal::v3::JournalTransactionInfoMarshaller	1799	activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMarshaller	156

activemq::wireformat::openwire::marshal::v4::ActiveMQByteMessageMarshaller	192	activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller	1501	activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller	1589
activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller	259	activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller	1589	activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller	1684
activemq::wireformat::openwire::marshal::v4::ActiveMQMapMessageMarshaller	294	activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller	1684	activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller	1727
activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller	317	activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller	1727	activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller	1760
activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMessageMarshaller	357	activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMessageMarshaller	1760	activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller	1775
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMessageMarshaller	392	activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMessageMarshaller	1775	activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller	1803
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMessageMarshaller	445	activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMessageMarshaller	1803	activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller	1829
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMessageMarshaller	467	activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMessageMarshaller	1829	activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller	1857
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMessageMarshaller	491	activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMessageMarshaller	1857	activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller	1888
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMessageMarshaller	515	activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMessageMarshaller	1888	activemq::wireformat::openwire::marshal::v4::LocalTransactionMarshaller	2074
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMessageMarshaller	539	activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMessageMarshaller	2074	activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller	2101
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMessageMarshaller	563	activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMessageMarshaller	2101	activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller	2126
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMessageMarshaller	619	activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMessageMarshaller	2126	activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller	2152
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMessageMarshaller	703	activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMessageMarshaller	2152	activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller	2172
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMessageMarshaller	731	activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMessageMarshaller	2172	activemq::wireformat::openwire::marshal::v4::MessageMarshaller	2225
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMessageMarshaller	1069	activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMessageMarshaller	2225	activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller	2267
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMessageMarshaller	1093	activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMessageMarshaller	2267	activemq::wireformat::openwire::marshal::v4::NetworkBridgeMarshaller	2333
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMessageMarshaller	1118	activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMessageMarshaller	2333	activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller	2433
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMessageMarshaller	1140	activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMessageMarshaller	2433	activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller	2467
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMessageMarshaller	1176	activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMessageMarshaller	2467	activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller	2488
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMessageMarshaller	1200	activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMessageMarshaller	2488	activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller	2558
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMessageMarshaller	1232	activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMessageMarshaller	2558	activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller	2582
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMessageMarshaller	1252	activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMessageMarshaller	2582	activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionMarshaller	2601
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMessageMarshaller	1277	activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMessageMarshaller	2601	activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller	2640
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMessageMarshaller	1324	activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMessageMarshaller	2640	activemq::wireformat::openwire::marshal::v4::ResponseMarshaller	2695
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMessageMarshaller	1405	activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMessageMarshaller	2695	activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller	2715
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMessageMarshaller	1429	activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMessageMarshaller	2715	activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller	

activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller	1208	activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller
2765		
activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller	1236	activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller
2925		
activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller	1260	activemq::wireformat::openwire::marshal::v5::ControlCommandMarshaller
3027		
activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller	1285	activemq::wireformat::openwire::marshal::v5::DataArrayResponseMarshaller
3054		
activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller	1320	activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller
3176		
activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller	1413	activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller
3207		
activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMessageMarshaller	1433	activemq::wireformat::openwire::marshal::v5::DiscoveryEventMarshaller
160		
activemq::wireformat::openwire::marshal::v5::ActiveMQByteMessageMarshaller	1505	activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller
196		
activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller	1593	activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller
263		
activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller	1688	activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller
298		
activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller	1739	activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller
321		
activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller	1764	activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller
361		
activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMessageMarshaller	1787	activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller
396		
activemq::wireformat::openwire::marshal::v5::ActiveMQStreamMessageMarshaller	1810	activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller
449		
activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller	1821	activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller
471		
activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller	1861	activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller
495		
activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller	1896	activemq::wireformat::openwire::marshal::v5::LocalTransactionMarshaller
519		
activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller	2066	activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller
543		
activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller	2113	activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller
567		
activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller	2138	activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller
625		
activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller	2159	activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller
707		
activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller	2180	activemq::wireformat::openwire::marshal::v5::MessageMarshaller
735		
activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller	2221	activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller
1073		
activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller	2259	activemq::wireformat::openwire::marshal::v5::NetworkBridgeFailureMarshaller
1097		
activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller	2341	activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller
1122		
activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller	2437	activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller
1148		
activemq::wireformat::openwire::marshal::v5::ConsumerControlMarshaller	2459	activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller
1184		

activemq::wireformat::openwire::marshal::v5::ProductionIdMarshaller,	2484	1968, 1969
activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller,	2546	
activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller,	2566	activemq::commands::ConnectionInfo,
activemq::wireformat::openwire::marshal::v5::ReplyToGroupInfoMarshaller,	2597	1134
activemq::wireformat::openwire::marshal::v5::ResponseMarshaller,	2631	Map
activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller,	2683	PrimitiveValueNode, 2402
activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller,	2719	mapValue
activemq::wireformat::openwire::marshal::v5::ShutdownMarshaller,	2777	PrimitiveValueNode::PrimitiveValue,
activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller,	2921	2396
activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller,	3031	SessionInfoMarshaller,
activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller,	3043	decaf::internal::io::StandardInputStream,
activemq::wireformat::openwire::marshal::v5::WireFormatIdMarshaller,	3168	2820
activemq::wireformat::openwire::marshal::v5::MarkBlockMarshaller,	3215	decaf::io::BlockingByteArrayInputStream,
activemq::wireformat::openwire::marshal::v5::BaseDataStreamMarshaller,	646	669
activemq::wireformat::openwire::marshal::v5::BaseDataStreamMarshaller,	646	decaf::io::BufferedInputStream, 749
activemq::wireformat::openwire::marshal::v5::BaseDataStreamMarshaller,	646	decaf::io::ByteArrayInputStream, 831
activemq::wireformat::openwire::marshal::v5::BaseDataStreamMarshaller,	646	decaf::io::FilterInputStream, 1531
activemq::wireformat::openwire::marshal::v5::BaseDataStreamMarshaller,	646	decaf::io::InputStream, 1631
activemq::wireformat::openwire::marshal::v5::BaseDataStreamMarshaller,	646	decaf::net::SocketInputStream, 2799
activemq::wireformat::openwire::marshal::v5::BaseDataStreamMarshaller,	646	decaf::util::logging::SimpleLogger, 2785
activemq::wireformat::openwire::marshal::v5::BaseDataStreamMarshaller,	646	decaf::util::logging, 117
activemq::wireformat::openwire::marshal::v5::BaseDataStreamMarshaller,	646	1992
activemq::wireformat::openwire::marshal::v5::BaseDataStreamMarshaller,	646	2820
activemq::wireformat::openwire::marshal::v5::BaseDataStreamMarshaller,	646	669
activemq::wireformat::openwire::marshal::v5::BaseDataStreamMarshaller,	646	decaf::io::BlockingByteArrayInputStream,
activemq::wireformat::openwire::marshal::v5::BaseDataStreamMarshaller,	646	decaf::io::BufferedInputStream, 750
activemq::wireformat::openwire::marshal::v5::BaseDataStreamMarshaller,	646	decaf::io::ByteArrayInputStream, 831
activemq::wireformat::openwire::marshal::v5::BaseDataStreamMarshaller,	646	decaf::io::FilterInputStream, 1531
activemq::wireformat::openwire::marshal::v5::BaseDataStreamMarshaller,	646	decaf::io::InputStream, 1631
activemq::wireformat::openwire::marshal::v5::BaseDataStreamMarshaller,	646	decaf::net::SocketInputStream, 2799
activemq::wireformat::openwire::marshal::v5::BaseDataStreamMarshaller,	646	2392
activemq::wireformat::openwire::marshal::v5::BaseDataStreamMarshaller,	646	2304
activemq::wireformat::openwire::marshal::v5::BaseDataStreamMarshaller,	646	682
activemq::wireformat::openwire::marshal::v5::BaseDataStreamMarshaller,	646	2885
activemq::wireformat::openwire::marshal::v5::BaseDataStreamMarshaller,	646	3150
activemq::wireformat::openwire::marshal::v5::BaseDataStreamMarshaller,	646	marshalledProperties
activemq::wireformat::openwire::marshal::v5::BaseDataStreamMarshaller,	646	activemq::commands::Message, 2035
activemq::wireformat::openwire::marshal::v5::BaseDataStreamMarshaller,	646	marshalledSize
activemq::wireformat::openwire::marshal::v5::BaseDataStreamMarshaller,	646	activemq::wireformat::openwire::utils::BooleanStream,
activemq::wireformat::openwire::marshal::v5::BaseDataStreamMarshaller,	646	682

- marshalPrimitive
 - activemq::wireformat::openwire::marshal::PrimitiveTypeMarshaller, 2392
- marshalPrimitiveList
 - activemq::wireformat::openwire::marshal::PrimitiveTypeMarshaller, 2393
- marshalPrimitiveMap
 - activemq::wireformat::openwire::marshal::PrimitiveTypeMarshaller, 2393
- masterBroker
 - activemq::commands::BrokerInfo, 721
- Math
 - decaf::lang::Math, 2001
- max
 - decaf::lang::Math, 2004, 2005
- MAX_RADIX
 - decaf::lang::Character, 921
- MAX_VALUE
 - decaf::lang::Byte, 784
 - decaf::lang::Character, 921
 - decaf::lang::Double, 1451
 - decaf::lang::Float, 1554
 - decaf::lang::Integer, 1666
 - decaf::lang::Long, 1947
 - decaf::lang::Short, 2738
- maximumPendingMessageLimit
 - activemq::commands::ConsumerInfo, 1222
- MemoryUsage
 - activemq::util::MemoryUsage, 2015
- MESSAGE
 - activemq::wireformat::stomp::StompCommandConstants, 2873
- Message
 - activemq::commands::Message, 2022
- message
 - activemq::cmsutil::CmsTemplate::ReceiveExecutor, 2526
 - activemq::commands::JournalTrace, 1769
 - activemq::commands::MessageDispatch, 2088
 - decaf::lang::Exception, 1482
- MessageAck
 - activemq::commands::MessageAck, 2056
- messageAck
 - activemq::commands::JournalQueueAck, 1721
- MessageAckMarshaller
 - activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller, 2077
 - activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller, 2061
 - activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller, 2069
- activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller, 2077
- activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller, 2065
- MessageAckMarshaller, 2065
- activemq::commands::MessageAck, 2060
- MessageDispatch
 - activemq::commands::MessageDispatch, 2085
- MessageDispatchChannel
 - activemq::core::MessageDispatchChannel, 2090
- MessageDispatchMarshaller
 - activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller, 2109
 - activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller, 2097
 - activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller, 2105
 - activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller, 2101
 - activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller, 2113
- MessageDispatchNotification
 - activemq::commands::MessageDispatchNotification, 2117
- MessageDispatchNotificationMarshaller
 - activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller, 2133
 - activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller, 2121
 - activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller, 2129
 - activemq::wireformat::openwire::marshal::v4::MessageDispatchNotificationMarshaller, 2125
 - activemq::wireformat::openwire::marshal::v5::MessageDispatchNotificationMarshaller, 2137
- MessageEOFException
 - cms::MessageEOFException, 2141
- MessageFormatException
 - cms::MessageFormatException, 2142
- MessageId
 - activemq::commands::MessageId, 2143
- messageId
 - activemq::commands::JournalTopicAck, 1746
 - activemq::commands::Message, 2035
 - activemq::commands::MessageDispatchNotification, 2120
 - activemq::commands::MessagePull, 2207
 - MessageIdMarshaller
 - activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller, 2163

- activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller, 784
- 2147
- decaf::lang::Character, 921
- activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller, 1451
- 2155
- decaf::lang::Float, 1554
- activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller, 1666
- 2151
- decaf::lang::Long, 1947
- activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller, 2738
- 2159
- MINUTES
- decaf::util::concurrent::TimeUnit, 3013
- MessageMarshaller
- activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller, 2184
- activemq::transport::mock::MockTransport, 2299
- activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller, 2167
- Mutex
- activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller, 2176
- decaf::util::concurrent::Mutex, 2240
- mutex
- activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller, 2172
- decaf::io::FilterOutputStream, 1543
- activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller, 2180
- decaf::util::concurrent::ConditionHandle, 1047
- MessageNotReadableException
- cms::MessageNotReadableException, 2188
- decaf::util::concurrent::MutexHandle, 2244
- MessageNotWriteableException
- cms::MessageNotWriteableException, 2189
- MutexHandle
- decaf::util::concurrent::MutexHandle, 2244
- MessagePropertyInterceptor
- activemq::wireformat::openwire::utils::MessagePropertyInterceptor, 2198
- decaf::util::UUID, 3145
- MessagePull
- NaN
- activemq::commands::MessagePull, 2204
- decaf::lang::Double, 1451
- MessagePullMarshaller
- decaf::lang::Float, 1554
- activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller, 2216
- decaf::util::concurrent::TimeUnit, 3013
- activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller, 2208
- decaf::lang::System, 2955
- activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller, 2212
- activemq::transport::failover::FailoverTransport, 2224
- activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller, 2224
- activemq::transport::IOTransport, 1712
- activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller, 2220
- activemq::transport::mock::MockTransport, 2233
- messageSequenceId
- activemq::transport::Transport, 3068
- activemq::commands::JournalTopicAck, 1746
- activemq::transport::TransportFilter, 3077
- NEGATIVE_INFINITY
- decaf::lang::Double, 1452
- MethodNames
- decaf::lang::Float, 1554
- activemq::commands::BrokerError::StackTraceElement, 2812
- NetworkBridgeFilter
- activemq::commands::NetworkBridgeFilter, 2247
- MICROSECONDS
- decaf::util::concurrent::TimeUnit, 3013
- MILLISECONDS
- decaf::util::concurrent::TimeUnit, 3013
- min
- decaf::lang::Math, 2005–2007
- MIN_RADIX
- decaf::lang::Character, 921
- activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilter, 2255
- MIN_VALUE
- 2255

- activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilterMarshaller, 2267
- activemq::wireformat::openwire::marshal::v5::NetworkBridgeFilterMarshaller, 2259
- networkBrokerId
 - activemq::commands::NetworkBridgeFilter, 2249
- networkConnection
 - activemq::commands::BrokerInfo, 721
- networkConsumerPath
 - activemq::commands::ConsumerInfo, 1222
- networkProperties
 - activemq::commands::BrokerInfo, 721
- networkSubscription
 - activemq::commands::ConsumerInfo, 1222
- networkTTL
 - activemq::commands::NetworkBridgeFilter, 2249
- newCondition
 - decaf::util::concurrent::locks::Lock, 1901
 - decaf::util::concurrent::locks::ReentrantLock, 2530
- newThread
 - decaf::util::concurrent::ThreadFactory, 2976
- next
 - activemq::transport::TransportFilter, 3080
 - decaf::util::Iterator, 1717
 - decaf::util::Random, 2515
- nextBoolean
 - decaf::util::Random, 2515
- nextBytes
 - decaf::util::Random, 2515
- nextDouble
 - decaf::util::Random, 2515
- nextFloat
 - decaf::util::Random, 2516
- nextGaussian
 - decaf::util::Random, 2516
- nextIndex
 - decaf::util::ListIterator, 1873
- nextInt
 - decaf::util::Random, 2516
- nextLong
 - decaf::util::Random, 2517
- nextToken
 - decaf::util::StringTokenizer, 2905, 2906
- node
 - decaf::util::UUID, 3145
- noLocal
 - activemq::cmsutil::CmsTemplate::ReceiveExecutor, 2526
 - activemq::commands::ConsumerInfo, 1222
- NON_PERSISTENT
- noRangeAcks
- normalize
 - decaf::net::URI, 3104
- NoRouteToHostException
 - decaf::net::NoRouteToHostException, 2270, 2271
- NoSuchAlgorithmException
 - decaf::security::NoSuchAlgorithmException, 2273
- NoSuchElementException
 - decaf::lang::exceptions::NoSuchElementException, 2275, 2276
- NoSuchProviderException
 - decaf::security::NoSuchProviderException, 2278
- notify
 - activemq::core::MessageDispatchChannel, 2092
 - decaf::internal::io::StandardErrorOutputStream, 2814
 - decaf::internal::io::StandardInputStream, 2821
 - decaf::internal::io::StandardOutputStream, 2827
 - decaf::internal::util::concurrent::ConditionImpl, 1048
 - decaf::internal::util::concurrent::SynchronizableImpl, 2943
 - decaf::io::BlockingByteArrayInputStream, 669
 - decaf::io::ByteArrayInputStream, 831
 - decaf::io::ByteArrayOutputStream, 838
 - decaf::io::FilterInputStream, 1531
 - decaf::io::FilterOutputStream, 1539
 - decaf::net::SocketInputStream, 2799
 - decaf::net::SocketOutputStream, 2805
 - decaf::util::AbstractCollection, 126
 - decaf::util::concurrent::ConcurrentStlMap, 1033
 - decaf::util::concurrent::Mutex, 2240
 - decaf::util::concurrent::Synchronizable, 2933
 - decaf::util::StlMap, 2853
 - decaf::util::StlQueue, 2862
- notifyAll
 - activemq::core::MessageDispatchChannel, 2092
 - decaf::internal::io::StandardErrorOutputStream, 2815
 - decaf::internal::io::StandardInputStream, 2821

- decaf::internal::io::StandardOutputStream, 2827
- decaf::internal::util::concurrent::ConditionImpl, 1049
- decaf::internal::util::concurrent::SynchronizableImpl, 2943
- decaf::io::BlockingByteArrayInputStream, 669
- decaf::io::ByteArrayInputStream, 831
- decaf::io::ByteArrayOutputStream, 839
- decaf::io::FilterInputStream, 1532
- decaf::io::FilterOutputStream, 1540
- decaf::net::SocketInputStream, 2799
- decaf::net::SocketOutputStream, 2806
- decaf::util::AbstractCollection, 126
- decaf::util::concurrent::ConcurrentStlMap, 1033
- decaf::util::concurrent::Mutex, 2241
- decaf::util::concurrent::Synchronizable, 2934
- decaf::util::StlMap, 2853
- decaf::util::StlQueue, 2862
- Null
 - decaf::util::logging, 117
- NULL_TYPE
 - activemq::util::PrimitiveValueNode, 2401
 - activemq::wireformat::openwire::OpenWireFormat, 2308
- NullPointerException
 - decaf::lang::exceptions::NullPointerException, 2280, 2281
- NUM_OPTIONS
 - activemq::core::ActiveMQConstants, 229
- NUM_PARAMS
 - activemq::core::ActiveMQConstants, 229
- NumberFormatException
 - decaf::lang::exceptions::NumberFormatException, 2285, 2286
- numberOfLeadingZeros
 - decaf::lang::Integer, 1659
 - decaf::lang::Long, 1940
- numberOfTrailingZeros
 - decaf::lang::Integer, 1659
 - decaf::lang::Long, 1941
- numWaiting
 - decaf::util::concurrent::ConditionHandle, 1047
- numWake
 - decaf::util::concurrent::ConditionHandle, 1047
- objectId
 - activemq::commands::RemoveInfo, 2540
- Off
 - decaf::util::logging, 117
 - offer
 - decaf::util::concurrent::SynchronousQueue, 2950
 - decaf::util::PriorityQueue, 2417
 - decaf::util::Queue, 2508
 - offset
 - decaf::internal::nio::CharArrayBuffer, 930
 - onCommand
 - activemq::core::ActiveMQConnection, 208
 - activemq::transport::correlator::ResponseCorrelator, 2617
 - activemq::transport::DefaultTransportListener, 1384
 - activemq::transport::failover::FailoverTransportListener, 1526
 - activemq::transport::inactivity::InactivityMonitor, 1626
 - activemq::transport::logging::LoggingTransport, 1921
 - activemq::transport::mock::InternalCommandListener, 1690
 - activemq::transport::TransportFilter, 3077
 - activemq::transport::TransportListener, 3081
 - activemq::wireformat::openwire::OpenWireFormatNegotiator, 2311
 - oneway
 - activemq::core::ActiveMQConnection, 209
 - activemq::core::ActiveMQSession, 415
 - activemq::transport::correlator::ResponseCorrelator, 2618
 - activemq::transport::failover::FailoverTransport, 1519
 - activemq::transport::inactivity::InactivityMonitor, 1626
 - activemq::transport::IOTransport, 1713
 - activemq::transport::logging::LoggingTransport, 1921
 - activemq::transport::mock::MockTransport, 2233
 - activemq::transport::Transport, 3069
 - activemq::transport::TransportFilter, 3077
 - activemq::wireformat::openwire::OpenWireFormatNegotiator, 2311
 - onException
 - activemq::core::ActiveMQConnection, 209
 - activemq::transport::DefaultTransportListener, 1385
 - activemq::transport::failover::BackupTransport, 593
 - activemq::transport::failover::FailoverTransportListener, 1526

- activemq::transport::inactivity::InactivityMonitor, 1626
- activemq::transport::TransportFilter, 3078
- activemq::transport::TransportListener, 3081
- cms::ExceptionListener, 1483
- onMessage
 - cms::MessageListener, 2166
- onProducerAck
 - activemq::core::ActiveMQProducer, 371
- onPropertyChanged
 - decaf::util::logging::PropertiesChangeListener, 2504
- onResponse
 - activemq::state::Tracked, 3015
- onSend
 - activemq::commands::ActiveMQBytesMessage, 172
 - activemq::commands::ActiveMQMessageTemplate, 338
 - activemq::commands::ActiveMQStreamMessage, 426
 - activemq::commands::Message, 2030
- onTaskComplete
 - decaf::util::concurrent::TaskListener, 2957
- onTaskCompleted
 - decaf::util::concurrent::PooledThreadListener, 2367
 - decaf::util::concurrent::ThreadPool, 2981
- onTaskException
 - decaf::util::concurrent::PooledThreadListener, 2367
 - decaf::util::concurrent::TaskListener, 2957
 - decaf::util::concurrent::ThreadPool, 2981
- onTaskStarted
 - decaf::util::concurrent::PooledThreadListener, 2367
 - decaf::util::concurrent::ThreadPool, 2981
- onTransportException
 - activemq::transport::correlator::ResponseCorrelator, 2618
 - activemq::wireformat::openwire::OpenWireFormatNegotiator, 2312
- OpenSSLX500Principal
 - decaf::security_-
 - provider::unix::openssl::OpenSSLX500Principal, 2288
- OpenWireFormat
 - activemq::wireformat::openwire::OpenWireFormat, 2299
- OpenWireFormatFactory
 - activemq::wireformat::openwire::OpenWireFormatFactory, 2309
- OpenWireFormatNegotiator
 - activemq::wireformat::openwire::OpenWireFormatNegotiator, 2310
 - OpenWireResponseBuilder
 - activemq::wireformat::openwire::OpenWireResponseBuilder, 2314
 - OpenwireStringSupport
 - activemq::wireformat::openwire::utils::OpenwireStringSupport, 2315
 - operationType
 - activemq::commands::DestinationInfo, 1395
 - activemq::commands::BrokerId, 693
 - activemq::commands::ConnectionId, 1108
 - activemq::commands::ConsumerId, 1194
 - activemq::commands::LocalTransactionId, 1877
 - activemq::commands::MessageId, 2145
 - activemq::commands::ProducerId, 2449
 - activemq::commands::SessionId, 2680
 - activemq::commands::TransactionId, 3018
 - activemq::commands::XATransactionId, 3196
 - decaf::lang::Boolean, 676
 - decaf::lang::Byte, 780, 781
 - decaf::lang::Character, 919, 920
 - decaf::lang::Comparable, 1010
 - decaf::lang::Double, 1448
 - decaf::lang::Float, 1550, 1551
 - decaf::lang::Integer, 1659, 1660
 - decaf::lang::Long, 1941
 - decaf::lang::Short, 2734
 - decaf::net::URI, 3104
 - decaf::nio::ByteBuffer, 864
 - decaf::nio::CharBuffer, 940
 - decaf::nio::DoubleBuffer, 1466
 - decaf::nio::FloatBuffer, 1569
 - decaf::nio::IntBuffer, 1648
 - decaf::nio::LongBuffer, 1962
 - decaf::nio::ShortBuffer, 2752
 - decaf::util::concurrent::TimeUnit, 3008
 - decaf::util::Date, 1380
 - decaf::util::UUID, 3145
 - operator*
 - decaf::lang::Pointer, 2347
 - operator()
 - decaf::lang::PointerComparator, 2351
 - decaf::util::Comparator, 1013
 - decaf::util::comparators::Less, 1864
 - std::less< decaf::lang::Pointer< T > >, 1865
 - operator=
 - decaf::lang::Pointer, 2348

- activemq::commands::BrokerId, 693
- activemq::commands::BrokerInfo, 719
- activemq::commands::ConnectionControl, 1058
- activemq::commands::ConnectionError, 1082
- activemq::commands::ConnectionId, 1108
- activemq::commands::ConnectionInfo, 1133
- activemq::commands::ConsumerControl, 1169
- activemq::commands::ConsumerId, 1194
- activemq::commands::ConsumerInfo, 1220
- activemq::commands::ControlCommand, 1245
- activemq::commands::DataArrayResponse, 1270
- activemq::commands::DataResponse, 1309
- activemq::commands::DestinationInfo, 1394
- activemq::commands::DiscoveryEvent, 1419
- activemq::commands::ExceptionResponse, 1486
- activemq::commands::FlushCommand, 1575
- activemq::commands::IntegerResponse, 1669
- activemq::commands::JournalQueueAck, 1721
- activemq::commands::JournalTopicAck, 1745
- activemq::commands::JournalTrace, 1769
- activemq::commands::JournalTransaction, 1792
- activemq::commands::KeepAliveInfo, 1814
- activemq::commands::LastPartialCommand, 1842
- activemq::commands::LocalTransactionId, 1877
- activemq::commands::Message, 2030
- activemq::commands::MessageAck, 2058
- activemq::commands::MessageDispatch, 2087
- activemq::commands::MessageDispatchNotification, 2118
- activemq::commands::MessageId, 2145
- activemq::commands::MessagePull, 2206
- activemq::commands::NetworkBridgeFilter, 2249
- activemq::commands::PartialCommand, 2321
- activemq::commands::ProducerAck, 2422
- activemq::commands::ProducerId, 2449
- activemq::commands::ProducerInfo, 2472
- activemq::commands::RemoveInfo, 2539
- activemq::commands::RemoveSubscriptionInfo, 2563
- activemq::commands::ReplayCommand, 2586
- activemq::commands::Response, 2613
- activemq::commands::SessionId, 2681
- activemq::commands::SessionInfo, 2704
- activemq::commands::ShutdownInfo, 2759
- activemq::commands::SubscriptionInfo, 2910
- activemq::commands::TransactionId, 3018
- activemq::commands::TransactionInfo, 3039
- activemq::commands::XATransactionId, 3196
- activemq::library::ActiveMQCPP, 239
- activemq::util::PrimitiveValueNode, 2408
- decaf::lang::Exception, 1481
- decaf::lang::Pointer, 2348
- decaf::util::AbstractCollection, 126
- decaf::util::Date, 1380
- decaf::util::logging::LogManager, 1927
- decaf::util::PriorityQueue, 2418
- decaf::util::Properties, 2500
- operator==
 - activemq::commands::BrokerId, 693
 - activemq::commands::ConnectionId, 1108
 - activemq::commands::ConsumerId, 1194
 - activemq::commands::LocalTransactionId, 1877
 - activemq::commands::MessageId, 2145
 - activemq::commands::ProducerId, 2449
 - activemq::commands::SessionId, 2681
 - activemq::commands::TransactionId, 3018
 - activemq::commands::XATransactionId, 3196
 - activemq::util::PrimitiveValueNode, 2408
 - decaf::lang, 107
 - decaf::lang::Boolean, 677
 - decaf::lang::Byte, 781
 - decaf::lang::Character, 920
 - decaf::lang::Comparable, 1011
 - decaf::lang::Double, 1448, 1449
 - decaf::lang::Float, 1551
 - decaf::lang::Integer, 1660
 - decaf::lang::Long, 1942
 - decaf::lang::Pointer, 2349, 2350
 - decaf::lang::Short, 2734, 2735
 - decaf::net::URI, 3105
 - decaf::nio::ByteBuffer, 865
 - decaf::nio::CharBuffer, 940
 - decaf::nio::DoubleBuffer, 1467

- decaf::nio::FloatBuffer, 1569
- decaf::nio::IntBuffer, 1648
- decaf::nio::LongBuffer, 1963
- decaf::nio::ShortBuffer, 2753
- decaf::util::concurrent::TimeUnit, 3009
- decaf::util::Date, 1380
- decaf::util::UUID, 3146
- operator[]
 - activemq::wireformat::openwire::utils::HexTable, 1610
 - decaf::internal::util::ByteArrayAdapter, 799
- optimizedAcknowledge
 - activemq::commands::ConsumerInfo, 1222
- options
 - activemq::commands::ActiveMQDestination, 249
- ordered
 - activemq::commands::ActiveMQDestination, 249
- orderedTarget
 - activemq::commands::ActiveMQDestination, 249
- originalDestination
 - activemq::commands::Message, 2035
- originalTransactionId
 - activemq::commands::Message, 2035
- outputStream
 - decaf::io::FilterOutputStream, 1543
- own
 - decaf::io::FilterInputStream, 1536
 - decaf::io::FilterOutputStream, 1543
- PARAM_CLIENTID
 - activemq::core::ActiveMQConstants, 229
- PARAM_PASSWORD
 - activemq::core::ActiveMQConstants, 229
- PARAM_USERNAME
 - activemq::core::ActiveMQConstants, 229
- parent
 - activemq::cmsutil::CmsTemplate::ProducerExecutor, 2446
 - activemq::cmsutil::CmsTemplate::ReceiveExecutor, 2526
- park
 - decaf::util::concurrent::locks::LockSupport, 1906
- parkNanos
 - decaf::util::concurrent::locks::LockSupport, 1906
- parkUntil
 - decaf::util::concurrent::locks::LockSupport, 1907
- parse
 - decaf::internal::util::HexStringParser, 1608
 - parseAuthority
 - decaf::internal::net::URIHelper, 3114
 - parseBoolean
 - decaf::lang::Boolean, 677
 - parseByte
 - decaf::lang::Byte, 782
 - parseComposite
 - activemq::util::URISupport, 3120
 - parseDouble
 - decaf::internal::util::HexStringParser, 1608
 - decaf::lang::Double, 1449
 - parseFloat
 - decaf::internal::util::HexStringParser, 1609
 - decaf::lang::Float, 1552
 - parseInt
 - decaf::lang::Integer, 1661
 - parseLong
 - decaf::lang::Long, 1942, 1943
 - parseQuery
 - activemq::util::URISupport, 3121
 - parseServerAuthority
 - decaf::net::URI, 3105
 - parseShort
 - decaf::lang::Short, 2735
 - parseURI
 - decaf::internal::net::URIHelper, 3114
 - parseURL
 - activemq::util::URISupport, 3121
 - PartialCommand
 - activemq::commands::PartialCommand, 2320
 - PartialCommandMarshaller
 - activemq::wireformat::openwire::marshal::v1::PartialCommand, 2336
 - activemq::wireformat::openwire::marshal::v2::PartialCommand, 2323
 - activemq::wireformat::openwire::marshal::v3::PartialCommand, 2327
 - activemq::wireformat::openwire::marshal::v4::PartialCommand, 2332
 - activemq::wireformat::openwire::marshal::v5::PartialCommand, 2340
 - password
 - activemq::commands::ConnectionInfo, 1134
 - peek
 - activemq::core::MessageDispatchChannel, 2092
 - decaf::internal::util::TimerTaskHeap, 3004
 - decaf::util::concurrent::SynchronousQueue, 2951
 - decaf::util::PriorityQueue, 2418
 - decaf::util::Queue, 2509

- peerBrokerInfos
 - activemq::commands::BrokerInfo, 721
- PERSISTENT
 - cms::DeliveryMode, 1387
- persistent
 - activemq::commands::Message, 2035
- physicalName
 - activemq::commands::ActiveMQDestination, 249
- PI
 - decaf::lang::Math, 2014
- Pointer
 - decaf::lang::Pointer, 2345, 2346
- PointerType
 - decaf::lang::Pointer, 2345
- poll
 - decaf::util::concurrent::SynchronousQueue, 2951
 - decaf::util::PriorityQueue, 2418
 - decaf::util::Queue, 2509
- PooledSession
 - activemq::cmsutil::PooledSession, 2354
- PooledThread
 - decaf::util::concurrent::PooledThread, 2364
- pop
 - decaf::util::StlQueue, 2862
- PortUnreachableException
 - decaf::net::PortUnreachableException, 2368, 2369
- position
 - decaf::nio::Buffer, 746
- POSITIVE_INFINITY
 - decaf::lang::Double, 1452
 - decaf::lang::Float, 1554
- pow
 - decaf::lang::Math, 2008
- prefetch
 - activemq::commands::ConsumerControl, 1170
- prefetchSize
 - activemq::commands::ConsumerInfo, 1222
- previous
 - decaf::util::ListIterator, 1873
- previousIndex
 - decaf::util::ListIterator, 1873
- PrimitiveList
 - activemq::util::PrimitiveList, 2373
- PrimitiveMap
 - activemq::util::PrimitiveMap, 2383
- PrimitiveType
 - activemq::util::PrimitiveValueNode, 2401
- PrimitiveTypesMarshaller
 - activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller, 2392
- PrimitiveValueConverter
 - activemq::util::PrimitiveValueConverter, 2397
- PrimitiveValueNode
 - activemq::util::PrimitiveValueNode, 2402–2404
- printStackTrace
 - cms::CMSEException, 963
 - decaf::lang::Exception, 1481
 - decaf::lang::Throwable, 2986
- priority
 - activemq::commands::ConsumerInfo, 1222
 - activemq::commands::Message, 2035
- PriorityQueue
 - decaf::util::PriorityQueue, 2415, 2416
- PriorityQueueIterator
 - decaf::util::PriorityQueue, 2420
- processBeginTransaction
 - activemq::state::CommandVisitor, 998
 - activemq::state::CommandVisitorAdapter, 1007
 - activemq::state::ConnectionStateTracker, 1162
- processBrokerError
 - activemq::state::CommandVisitor, 998
 - activemq::state::CommandVisitorAdapter, 1007
- processBrokerInfo
 - activemq::state::CommandVisitor, 999
 - activemq::state::CommandVisitorAdapter, 1007
- processCommitTransactionOnePhase
 - activemq::state::CommandVisitor, 999
 - activemq::state::CommandVisitorAdapter, 1007
 - activemq::state::ConnectionStateTracker, 1162
- processCommitTransactionTwoPhase
 - activemq::state::CommandVisitor, 999
 - activemq::state::CommandVisitorAdapter, 1007
 - activemq::state::ConnectionStateTracker, 1162
- processConnectionControl
 - activemq::state::CommandVisitor, 999
 - activemq::state::CommandVisitorAdapter, 1007
- processConnectionError
 - activemq::state::CommandVisitor, 999
 - activemq::state::CommandVisitorAdapter, 1007
- processTypesMarshaller
 - activemq::state::CommandVisitor, 999

- activemq::state::CommandVisitorAdapter, 1007
- activemq::state::ConnectionStateTracker, 1162
- processConsumerControl
 - activemq::state::CommandVisitor, 999
 - activemq::state::CommandVisitorAdapter, 1007
- processConsumerInfo
 - activemq::state::CommandVisitor, 999
 - activemq::state::CommandVisitorAdapter, 1007
 - activemq::state::ConnectionStateTracker, 1163
- processControlCommand
 - activemq::state::CommandVisitor, 999
 - activemq::state::CommandVisitorAdapter, 1007
- processDestinationInfo
 - activemq::state::CommandVisitor, 1000
 - activemq::state::CommandVisitorAdapter, 1007
 - activemq::state::ConnectionStateTracker, 1163
- processEndTransaction
 - activemq::state::CommandVisitor, 1000
 - activemq::state::CommandVisitorAdapter, 1007
 - activemq::state::ConnectionStateTracker, 1163
- processFlushCommand
 - activemq::state::CommandVisitor, 1000
 - activemq::state::CommandVisitorAdapter, 1007
- processForgetTransaction
 - activemq::state::CommandVisitor, 1000
 - activemq::state::CommandVisitorAdapter, 1007
- processKeepAliveInfo
 - activemq::state::CommandVisitor, 1000
 - activemq::state::CommandVisitorAdapter, 1007
- processMessage
 - activemq::state::CommandVisitor, 1000
 - activemq::state::CommandVisitorAdapter, 1007
 - activemq::state::ConnectionStateTracker, 1163
- processMessageAck
 - activemq::state::CommandVisitor, 1000
 - activemq::state::CommandVisitorAdapter, 1007
 - activemq::state::ConnectionStateTracker, 1163
- processMessageDispatch
 - activemq::state::CommandVisitor, 1000
 - activemq::state::CommandVisitorAdapter, 1007
- processMessageDispatchNotification
 - activemq::state::CommandVisitor, 1001
 - activemq::state::CommandVisitorAdapter, 1007
- processMessagePull
 - activemq::state::CommandVisitor, 1001
 - activemq::state::CommandVisitorAdapter, 1007
- processPrepareTransaction
 - activemq::state::CommandVisitor, 1001
 - activemq::state::CommandVisitorAdapter, 1007
 - activemq::state::ConnectionStateTracker, 1163
- processProducerAck
 - activemq::state::CommandVisitor, 1001
 - activemq::state::CommandVisitorAdapter, 1007
- processProducerInfo
 - activemq::state::CommandVisitor, 1001
 - activemq::state::CommandVisitorAdapter, 1007
 - activemq::state::ConnectionStateTracker, 1163
- processRecoverTransactions
 - activemq::state::CommandVisitor, 1001
 - activemq::state::CommandVisitorAdapter, 1007
- processRemoveConnection
 - activemq::state::CommandVisitor, 1001
 - activemq::state::CommandVisitorAdapter, 1007
 - activemq::state::ConnectionStateTracker, 1164
- processRemoveConsumer
 - activemq::state::CommandVisitor, 1001
 - activemq::state::CommandVisitorAdapter, 1007
 - activemq::state::ConnectionStateTracker, 1164
- processRemoveDestination
 - activemq::state::CommandVisitor, 1001
 - activemq::state::CommandVisitorAdapter, 1007
 - activemq::state::ConnectionStateTracker, 1164
- processRemoveInfo
 - activemq::state::CommandVisitor, 1002
 - activemq::state::CommandVisitorAdapter, 1007

- processRemoveProducer
 - activemq::state::CommandVisitor, 1002
 - activemq::state::CommandVisitorAdapter, 1008
 - activemq::state::ConnectionStateTracker, 1164
- processRemoveSession
 - activemq::state::CommandVisitor, 1002
 - activemq::state::CommandVisitorAdapter, 1008
 - activemq::state::ConnectionStateTracker, 1164
- processRemoveSubscriptionInfo
 - activemq::state::CommandVisitor, 1002
 - activemq::state::CommandVisitorAdapter, 1008
- processReplayCommand
 - activemq::state::CommandVisitor, 1002
 - activemq::state::CommandVisitorAdapter, 1008
- processResponse
 - activemq::state::CommandVisitor, 1002
 - activemq::state::CommandVisitorAdapter, 1008
- processRollbackTransaction
 - activemq::state::CommandVisitor, 1002
 - activemq::state::CommandVisitorAdapter, 1008
 - activemq::state::ConnectionStateTracker, 1164
- processSessionInfo
 - activemq::state::CommandVisitor, 1002
 - activemq::state::CommandVisitorAdapter, 1008
 - activemq::state::ConnectionStateTracker, 1164
- processShutdownInfo
 - activemq::state::CommandVisitor, 1003
 - activemq::state::CommandVisitorAdapter, 1008
- processTransactionInfo
 - activemq::state::CommandVisitor, 1003
 - activemq::state::CommandVisitorAdapter, 1008
- processWireFormat
 - activemq::state::CommandVisitor, 1003
 - activemq::state::CommandVisitorAdapter, 1009
- PRODUCER_ADVISORY_PREFIX
 - activemq::commands::ActiveMQDestination, 249
- ProducerAck
 - activemq::commands::ProducerAck, 2421
- ProducerAckMarshaller
 - activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller, 2441
 - activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller, 2425
 - activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller, 2429
 - activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller, 2433
 - activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller, 2437
- ProducerExecutor
 - activemq::cmsutil::CmsTemplate, 981
 - activemq::cmsutil::CmsTemplate::ProducerExecutor, 2445
- ProducerId
 - activemq::commands::ProducerId, 2447
- producerId
 - activemq::commands::Message, 2035
 - activemq::commands::MessageId, 2146
 - activemq::commands::ProducerAck, 2423
 - activemq::commands::ProducerInfo, 2474
- ProducerIdMarshaller
 - activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller, 2463
 - activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller, 2451
 - activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller, 2455
 - activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller, 2467
 - activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller, 2459
- ProducerInfo
 - activemq::commands::ProducerInfo, 2471
- ProducerInfoMarshaller
 - activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller, 2491
 - activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller, 2475
 - activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller, 2479
 - activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller, 2487
 - activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller, 2483
- producerSequenceId
 - activemq::commands::MessageId, 2146
- ProducerState
 - activemq::state::ProducerState, 2494
- Properties
 - decaf::util::Properties, 2496
- propertyExists
 - activemq::commands::ActiveMQMessageTemplate, 338

- cms::Message, 2048
- ProtocolException
 - decaf::net::ProtocolException, 2505, 2506
- publish
 - decaf::util::logging::Handler, 1606
 - decaf::util::logging::StreamHandler, 2891
- purge
 - decaf::util::Timer, 2992
- push
 - decaf::util::StlQueue, 2863
- put
 - decaf::internal::nio::ByteBuffer, 821
 - decaf::internal::nio::CharArrayBuffer, 928
 - decaf::internal::nio::DoubleArrayBuffer, 1458
 - decaf::internal::nio::FloatArrayBuffer, 1560, 1561
 - decaf::internal::nio::IntArrayBuffer, 1640
 - decaf::internal::nio::LongArrayBuffer, 1954
 - decaf::internal::nio::ShortArrayBuffer, 2744
 - decaf::internal::util::ByteArrayAdapter, 800
 - decaf::nio::ByteBuffer, 865–867
 - decaf::nio::CharBuffer, 940–943
 - decaf::nio::DoubleBuffer, 1467–1469
 - decaf::nio::FloatBuffer, 1569–1571
 - decaf::nio::IntBuffer, 1649, 1650
 - decaf::nio::LongBuffer, 1963–1965
 - decaf::nio::ShortBuffer, 2753–2755
 - decaf::util::concurrent::ConcurrentStlMap, 1033
 - decaf::util::concurrent::SynchronousQueue, 2951
 - decaf::util::Map, 1978
 - decaf::util::StlMap, 2854
- putAll
 - decaf::util::concurrent::ConcurrentStlMap, 1034
 - decaf::util::Map, 1979
 - decaf::util::StlMap, 2854
- putChar
 - decaf::internal::nio::ByteBuffer, 822
 - decaf::internal::util::ByteArrayAdapter, 800
 - decaf::nio::ByteBuffer, 867, 868
- putDouble
 - decaf::internal::nio::ByteBuffer, 823
 - decaf::internal::util::ByteArrayAdapter, 800
 - decaf::nio::ByteBuffer, 868
- putDoubleAt
 - decaf::internal::util::ByteArrayAdapter, 801
- putFloat
 - decaf::internal::nio::ByteBuffer, 823, 824
 - decaf::internal::util::ByteArrayAdapter, 801
 - decaf::nio::ByteBuffer, 869
- putFloatAt
 - decaf::internal::util::ByteArrayAdapter, 802
- putIfAbsent
 - decaf::util::concurrent::ConcurrentMap, 1021
 - decaf::util::concurrent::ConcurrentStlMap, 1034
- putInt
 - decaf::internal::nio::ByteBuffer, 824, 825
 - decaf::internal::util::ByteArrayAdapter, 802
 - decaf::nio::ByteBuffer, 870
- putIntAt
 - decaf::internal::util::ByteArrayAdapter, 802
- putLong
 - decaf::internal::nio::ByteBuffer, 825, 826
 - decaf::internal::util::ByteArrayAdapter, 803
 - decaf::nio::ByteBuffer, 871
- putLongAt
 - decaf::internal::util::ByteArrayAdapter, 803
- putShort
 - decaf::internal::nio::ByteBuffer, 826
 - decaf::internal::util::ByteArrayAdapter, 804
 - decaf::nio::ByteBuffer, 871, 872
- putShortAt
 - decaf::internal::util::ByteArrayAdapter, 804
- QUEUE
 - cms::Destination, 1388
- QUEUE_PREFIX
 - activemq::wireformat::stomp::StompCommandConstants, 2873
- QUEUE_QUALIFIED_PREFIX
 - activemq::commands::ActiveMQDestination, 249
- queueTask
 - decaf::util::concurrent::ThreadPool, 2981
- quoteIllegal
 - decaf::internal::net::URIEncoderDecoder, 3109

- Random
 - decaf::util::Random, 2514
- random
 - decaf::lang::Math, 2008
- randomUUID
 - decaf::util::UUID, 3146
- read
 - activemq::io::LoggingInputStream, 1918
 - decaf::internal::io::StandardInputStream, 2821, 2822
 - decaf::internal::util::ByteArrayAdapter, 804
 - decaf::io::BlockingByteArrayInputStream, 670
 - decaf::io::BufferedInputStream, 750
 - decaf::io::ByteArrayInputStream, 832
 - decaf::io::DataInputStream, 1293, 1294
 - decaf::io::FilterInputStream, 1532, 1533
 - decaf::io::InputStream, 1632
 - decaf::io::Reader, 2519
 - decaf::net::SocketInputStream, 2800
 - decaf::nio::CharBuffer, 944
- readAsciiString
 - activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller, 648
- readBoolean
 - activemq::commands::ActiveMQBytesMessage, 172
 - activemq::commands::ActiveMQStreamMessage, 426
 - activemq::wireformat::openwire::utils::BooleanStream, 682
 - cms::BytesMessage, 878
 - cms::StreamMessage, 2895
 - decaf::io::DataInputStream, 1295
- readByte
 - activemq::commands::ActiveMQBytesMessage, 172
 - activemq::commands::ActiveMQStreamMessage, 427
 - cms::BytesMessage, 878
 - cms::StreamMessage, 2895
 - decaf::io::DataInputStream, 1295
 - decaf::io::Reader, 2519
- readBytes
 - activemq::commands::ActiveMQBytesMessage, 173
 - activemq::commands::ActiveMQStreamMessage, 427, 428
 - cms::BytesMessage, 879
 - cms::StreamMessage, 2896
- readChar
 - activemq::commands::ActiveMQBytesMessage, 174
- activemq::commands::ActiveMQStreamMessage, 428
- cms::BytesMessage, 880
- cms::StreamMessage, 2897
- decaf::io::DataInputStream, 1295
- ReadChecker
 - activemq::transport::inactivity::InactivityMonitor, 1627
 - activemq::transport::inactivity::ReadChecker, 2518
- readDouble
 - activemq::commands::ActiveMQBytesMessage, 174
 - activemq::commands::ActiveMQStreamMessage, 429
 - cms::BytesMessage, 880
 - cms::StreamMessage, 2897
 - decaf::io::DataInputStream, 1295
- readFloat
 - activemq::commands::ActiveMQBytesMessage, 174
 - activemq::commands::ActiveMQStreamMessage, 429
 - cms::BytesMessage, 880
 - cms::StreamMessage, 2898
 - decaf::io::DataInputStream, 1296
- readFully
 - decaf::io::DataInputStream, 1296, 1297
- readInt
 - activemq::commands::ActiveMQBytesMessage, 175
 - activemq::commands::ActiveMQStreamMessage, 429
 - cms::BytesMessage, 881
 - cms::StreamMessage, 2898
 - decaf::io::DataInputStream, 1297
- readLock
 - decaf::util::concurrent::locks::ReadWriteLock, 2524
- readLong
 - activemq::commands::ActiveMQBytesMessage, 175
 - activemq::commands::ActiveMQStreamMessage, 430
 - cms::BytesMessage, 881
 - cms::StreamMessage, 2898
 - decaf::io::DataInputStream, 1297
- readOnly
 - decaf::internal::nio::CharArrayBuffer, 930
- ReadOnlyBufferException
 - decaf::nio::ReadOnlyBufferException, 2520, 2521
- readShort

- activemq::commands::ActiveMQBytesMessage, 175
- activemq::commands::ActiveMQStreamMessage, 430
- cms::BytesMessage, 881
- cms::StreamMessage, 2899
- decaf::io::DataInputStream, 1298
- readString
 - activemq::commands::ActiveMQBytesMessage, 176
 - activemq::commands::ActiveMQStreamMessage, 430
 - activemq::wireformat::openwire::utils::OpenwireStreamSupport, 2315
 - cms::BytesMessage, 882
 - cms::StreamMessage, 2899
 - decaf::io::DataInputStream, 1298
- readUnsignedByte
 - decaf::io::DataInputStream, 1298
- readUnsignedShort
 - activemq::commands::ActiveMQBytesMessage, 176
 - activemq::commands::ActiveMQStreamMessage, 431
 - cms::BytesMessage, 882
 - cms::StreamMessage, 2900
 - decaf::io::DataInputStream, 1299
- readUTF
 - activemq::commands::ActiveMQBytesMessage, 176
 - cms::BytesMessage, 882
 - decaf::io::DataInputStream, 1299
- RECEIPT
 - activemq::wireformat::stomp::StompCommandConstants, 2873
- receive
 - activemq::cmsutil::CachedConsumer, 889, 890
 - activemq::cmsutil::CmsTemplate, 976, 977
 - activemq::core::ActiveMQConsumer, 236
 - cms::MessageConsumer, 2081, 2082
- RECEIVE_TIMEOUT_INDEFINITE_WAIT
 - activemq::cmsutil::CmsTemplate, 981
- RECEIVE_TIMEOUT_NO_WAIT
 - activemq::cmsutil::CmsTemplate, 982
- ReceiveExecutor
 - activemq::cmsutil::CmsTemplate, 981
 - activemq::cmsutil::CmsTemplate::ReceiveExecutor, 2525
- receiveNoWait
 - activemq::cmsutil::CachedConsumer, 890
 - activemq::core::ActiveMQConsumer, 236
 - cms::MessageConsumer, 2082
- receiveSelected
 - activemq::cmsutil::CmsTemplate, 977, 978
 - recievedByDFBridge
 - activemq::commands::Message, 2035
 - reconnect
 - activemq::transport::failover::FailoverTransport, 1519
 - activemq::transport::IOTransport, 1713
 - activemq::transport::mock::MockTransport, 2233
 - activemq::transport::Transport, 3069
 - activemq::transport::TransportFilter, 3078
 - recover
 - activemq::cmsutil::PooledSession, 2362
 - activemq::core::ActiveMQSession, 415
 - cms::Session, 2675
 - redeliveryCounter
 - activemq::commands::Message, 2035
 - activemq::commands::MessageDispatch, 2088
 - redispatch
 - activemq::core::ActiveMQSession, 416
 - ReentrantLock
 - decaf::util::concurrent::locks::ReentrantLock, 2528
 - ReferenceType
 - decaf::lang::Pointer, 2345
 - registerFactory
 - activemq::transport::TransportRegistry, 3084
 - activemq::wireformat::WireFormatRegistry, 3185
 - registerSecurityProvider
 - decaf::security_ -
 - activemq::provider::SecurityProviderMap, 2649
 - RejectedExecutionException
 - decaf::util::concurrent::RejectedExecutionException, 2534, 2535
 - relativize
 - decaf::net::URI, 3105
 - release
 - decaf::lang::AtomicRefCounter, 588
 - decaf::lang::Pointer, 2349
 - decaf::util::concurrent::Semaphore, 2656, 2657
 - releaseAll
 - activemq::cmsutil::ResourceLifecycleManager, 2610
 - remaining
 - decaf::nio::Buffer, 746
 - remainingCapacity
 - decaf::util::concurrent::SynchronousQueue, 2952
 - remove
 - activemq::util::ActiveMQProperties, 378

- cms::CMSProperties, 967
- decaf::internal::util::TimerTaskHeap, 3004
- decaf::util::AbstractCollection, 127
- decaf::util::AbstractQueue, 136
- decaf::util::Collection, 988
- decaf::util::concurrent::ConcurrentMap, 1022
- decaf::util::concurrent::ConcurrentStlMap, 1035, 1036
- decaf::util::concurrent::SynchronousQueue, 2952
- decaf::util::Iterator, 1717
- decaf::util::List, 1870
- decaf::util::Map, 1980
- decaf::util::PriorityQueue, 2418, 2419
- decaf::util::Properties, 2500
- decaf::util::Queue, 2510
- decaf::util::StlList, 2843
- decaf::util::StlMap, 2854
- decaf::util::StlSet, 2870
- removeAll
 - activemq::core::MessageDispatchChannel, 2093
 - decaf::util::AbstractCollection, 127
 - decaf::util::AbstractSet, 139
 - decaf::util::Collection, 988
 - decaf::util::concurrent::SynchronousQueue, 2952
- removeConsumer
 - activemq::state::SessionState, 2728
- removeDispatcher
 - activemq::core::ActiveMQConnection, 209
- removeHandler
 - decaf::util::logging::Logger, 1915
- RemoveInfo
 - activemq::commands::RemoveInfo, 2538
- RemoveInfoMarshaller
 - activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller, 2541
 - activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller, 2549
 - activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller, 2553
 - activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller, 2557
 - activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller, 2545
- removeProducer
 - activemq::core::ActiveMQConnection, 209
 - activemq::state::SessionState, 2728
- removeProperty
 - activemq::wireformat::stomp::StompFrame, 2877
- removePropertyChangeListener
 - decaf::util::logging::LogManager, 1927
- removeSession
 - activemq::core::ActiveMQConnection, 209
 - activemq::state::ConnectionState, 1159
- RemoveSubscriptionInfo
 - activemq::commands::RemoveSubscriptionInfo, 2561
- RemoveSubscriptionInfoMarshaller
 - activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller, 2573
 - activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller, 2569
 - activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller, 2577
 - activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionInfoMarshaller, 2581
 - activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller, 2565
- removeSynchronization
 - activemq::core::ActiveMQTransactionContext, 576
- removeTask
 - activemq::threads::CompositeTaskRunner, 1018
- removeTempDestination
 - activemq::state::ConnectionState, 1159
- RemoveTransactionAction
 - activemq::state::ConnectionStateTracker, 1165
- removeTransactionState
 - activemq::state::ConnectionState, 1159
- removeTransportListener
 - activemq::core::ActiveMQConnection, 210
- removeURI
 - activemq::transport::CompositeTransport, 1020
 - activemq::transport::failover::FailoverTransport, 1154
 - activemq::transport::failover::URIPool, 1154
 - renegotiateWireFormat
 - activemq::wireformat::openwire::OpenWireFormat, 2304
- ReplayCommand
 - decaf::util::concurrent::ConcurrentMap, 1028
 - decaf::util::concurrent::ConcurrentStlMap, 1036, 1037
- ReplayCommandMarshaller
 - activemq::commands::ReplayCommand, 2585
 - activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller, 2604

- activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller, 2588
- activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller, 2592
- activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller, 2600
- activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller, 2596
- activemq::cmsutil::CmsTemplate, 981
- activemq::cmsutil::CmsTemplate::ResolveProducerExecutor, 2607
- ResolveReceiveExecutor
- activemq::cmsutil::CmsTemplate, 981
- activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor, 2608
- ResourceLifecycleManager
- activemq::cmsutil::ResourceLifecycleManager, 2609
- Response
- activemq::commands::Response, 2612
- ResponseCorrelator
- activemq::transport::correlator::ResponseCorrelator, 2617
- ResponseMarshaller
- activemq::wireformat::openwire::marshal::v1::ResponseMarshaller, 2634
- activemq::wireformat::openwire::marshal::v2::ResponseMarshaller, 2621
- activemq::wireformat::openwire::marshal::v3::ResponseMarshaller, 2625
- activemq::wireformat::openwire::marshal::v4::ResponseMarshaller, 2639
- activemq::wireformat::openwire::marshal::v5::ResponseMarshaller, 2630
- restore
- activemq::state::ConnectionStateTracker, 1165
- restoreTransport
- activemq::transport::failover::FailoverTransport, 1520
- result
- activemq::commands::IntegerResponse, 1669
- resume
- activemq::commands::ConnectionControl, 1059
- retainAll
- decaf::util::AbstractCollection, 128
- decaf::util::Collection, 989
- decaf::util::concurrent::SynchronousQueue, 2952
- retroactive
- activemq::commands::ConsumerInfo, 1222
- returnInstance
- decaf::util::logging::LogManager, 1927
- activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller, 2588
- activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller, 2592
- activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller, 2600
- activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller, 2596
- replyTo
- activemq::commands::Message, 2035
- request
- activemq::transport::correlator::ResponseCorrelator, 2618, 2619
- activemq::transport::failover::FailoverTransport, 1519, 1520
- activemq::transport::IOTransport, 1713
- activemq::transport::logging::LoggingTransport, 1922
- activemq::transport::mock::MockTransport, 2233, 2234
- activemq::transport::Transport, 3069, 3070
- activemq::transport::TransportFilter, 3078, 3079
- activemq::wireformat::openwire::OpenWireFormatNegotiator, 2312
- reserve
- decaf::util::concurrent::ThreadPool, 2982
- reset
- activemq::commands::ActiveMQBytesMessage, 177
- activemq::commands::ActiveMQStreamMessage, 431
- activemq::state::ConnectionState, 1160
- cms::BytesMessage, 883
- decaf::internal::io::StandardInputStream, 2822
- decaf::internal::util::TimerTaskHeap, 3004
- decaf::io::BlockingByteArrayInputStream, 670
- decaf::io::BufferedInputStream, 751
- decaf::io::ByteArrayInputStream, 832
- decaf::io::ByteArrayOutputStream, 839
- decaf::io::FilterInputStream, 1533
- decaf::io::InputStream, 1633
- decaf::lang::Pointer, 2349
- decaf::net::SocketInputStream, 2800
- decaf::nio::Buffer, 746
- decaf::util::StringTokenizer, 2906
- resize
- decaf::internal::util::ByteArrayAdapter, 805
- resolve
- decaf::net::URI, 3106
- resolveDestinationName

- decaf::util::logging::LogWriter, 1933
- returnRef
 - decaf::internal::nio::ByteArrayPerspective, 847
- returnSession
 - activemq::cmsutil::SessionPool, 2726
- reverse
 - decaf::lang::Integer, 1661
 - decaf::lang::Long, 1943
 - decaf::util::StlQueue, 2863
- reverseBytes
 - decaf::lang::Integer, 1662
 - decaf::lang::Long, 1943
 - decaf::lang::Short, 2736
- rewind
 - decaf::nio::Buffer, 747
- rollback
 - activemq::cmsutil::PooledSession, 2363
 - activemq::core::ActiveMQConsumer, 237
 - activemq::core::ActiveMQSession, 416
 - activemq::core::ActiveMQTransactionContext, 576
 - cms::Session, 2675
- rotateLeft
 - decaf::lang::Integer, 1662
 - decaf::lang::Long, 1943
- rotateRight
 - decaf::lang::Integer, 1662
 - decaf::lang::Long, 1944
- round
 - decaf::lang::Math, 2009
- run
 - activemq::threads::CompositeTaskRunner, 1018
 - activemq::threads::DedicatedTaskRunner, 1383
 - activemq::transport::inactivity::ReadChecker, 2518
 - activemq::transport::inactivity::WriteChecker, 3186
 - activemq::transport::IOTransport, 1714
 - activemq::transport::mock::InternalCommandListener, 1690
 - decaf::lang::Runnable, 2642
 - decaf::util::concurrent::PooledThread, 2365
- RuntimeException
 - decaf::lang::exceptions::RuntimeException, 2645, 2646
- schedule
 - decaf::util::Timer, 2992–2997
- scheduleAtFixedRate
 - decaf::util::Timer, 2997–2999
- scheduledExecutionTime
 - decaf::util::TimerTask, 3002
- SECONDS
 - decaf::util::concurrent::TimeUnit, 3013
- SecurityProviderRegistrar
 - decaf::security_ - provider::SecurityProviderRegistrar, 2650
- selector
 - activemq::cmsutil::CmsTemplate::ReceiveExecutor, 2526
 - activemq::commands::ConsumerInfo, 1222
 - activemq::commands::SubscriptionInfo, 2911
- Semaphore
 - decaf::util::concurrent::Semaphore, 2653, 2654
- semaphore
 - decaf::util::concurrent::ConditionHandle, 1047
- SEND
 - activemq::wireformat::stomp::StompCommandConstants, 2873
- send
 - activemq::cmsutil::CachedProducer, 894, 895
 - activemq::cmsutil::CmsTemplate, 978
 - activemq::core::ActiveMQProducer, 371, 372
 - activemq::core::ActiveMQSession, 416
 - cms::MessageProducer, 2192–2194
- SendExecutor
 - activemq::cmsutil::CmsTemplate, 981
 - activemq::cmsutil::CmsTemplate::SendExecutor, 2661
- sendPullRequest
 - activemq::core::ActiveMQConnection, 210
- ServerSocket
 - decaf::net::ServerSocket, 2662
- serviceName
 - activemq::commands::DiscoveryEvent, 1420
- SESSION_TRANSACTED
 - cms::Session, 2667
- SessionId
 - activemq::commands::SessionId, 2679
- sessionId
 - activemq::commands::ConsumerId, 1194
 - activemq::commands::ProducerId, 2449
 - activemq::commands::SessionInfo, 2705
- SessionIdMarshaller
 - activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller, 2687
 - activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller, 2691

- activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller, 2699
- activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller, 2695
- activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller, 2683
- SessionInfo
 - activemq::commands::SessionInfo, 2703
- SessionInfoMarshaller
 - activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller, 2710
 - activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller, 2706
 - activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller, 2722
 - activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller, 2714
 - activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller, 2718
- SessionPool
 - activemq::cmsutil::SessionPool, 2725
- SessionState
 - activemq::state::SessionState, 2728
- set
 - decaf::util::concurrent::atomic::AtomicBoolean, 581
 - decaf::util::concurrent::atomic::AtomicInteger, 586
 - decaf::util::concurrent::atomic::AtomicReference, 591
 - decaf::util::List, 1871
 - decaf::util::ListIterator, 1874
 - decaf::util::StlList, 2844
- setAbsolute
 - decaf::internal::net::URIType, 3130
- setAckHandler
 - activemq::commands::Message, 2031
- setAckMode
 - activemq::commands::SessionInfo, 2704
- setAckType
 - activemq::commands::MessageAck, 2059
- setAdditionalPredicate
 - activemq::commands::ConsumerInfo, 1220
- setAdvisory
 - activemq::commands::ActiveMQDestination, 247
- setAlwaysSyncSend
 - activemq::core::ActiveMQConnectionSupport, 225
- setArrival
 - activemq::commands::Message, 2031
- setAuthority
 - decaf::internal::net::URIType, 3130
- setBackOffMultiplier
 - activemq::transport::failover::FailoverTransport, 1520
 - activemq::transport::failover::FailoverTransport, 1521
 - activemq::transport::failover::FailoverTransport, 1521
 - setBackupPoolSize
 - activemq::transport::failover::BackupTransportPool, 596
 - activemq::transport::failover::FailoverTransport, 1521
 - setBlockSize
 - decaf::util::concurrent::ThreadPool, 2982
 - setBody
 - activemq::wireformat::stomp::StompFrame, 2877
 - setBodyBytes
 - activemq::commands::ActiveMQBytesMessage, 1977
 - cms::BytesMessage, 883
 - setBool
 - activemq::util::PrimitiveList, 2377
 - activemq::util::PrimitiveMap, 2388
 - activemq::util::PrimitiveValueNode, 2408
 - setBoolean
 - activemq::commands::ActiveMQMapMessage, 281
 - cms::MapMessage, 1988
 - setBooleanProperty
 - activemq::commands::ActiveMQMessageTemplate, 338
 - activemq::wireformat::openwire::utils::MessagePropertyInterce, 2200
 - cms::Message, 2048
 - setBranchQualifier
 - activemq::commands::XATransactionId, 3196
 - setBrokerId
 - activemq::commands::BrokerInfo, 719
 - setBrokerInTime
 - activemq::commands::Message, 2032
 - setBrokerMasterConnector
 - activemq::commands::ConnectionInfo, 1133
 - setBrokerName
 - activemq::commands::BrokerInfo, 719
 - activemq::commands::DiscoveryEvent, 1419
 - setBrokerOutTime
 - activemq::commands::Message, 2032
 - setBrokerPath
 - activemq::commands::ConnectionInfo, 1133
 - activemq::commands::ConsumerInfo, 1220

- activemq::commands::DestinationInfo, 1394
- activemq::commands::Message, 2032
- activemq::commands::ProducerInfo, 2473
- setBrokerSequenceId
 - activemq::commands::MessageId, 2145
- setBrokerUploadUrl
 - activemq::commands::BrokerInfo, 719
- setBrokerURL
 - activemq::commands::BrokerInfo, 719
 - activemq::core::ActiveMQConnectionFactory, 215
- setBrowser
 - activemq::commands::ConsumerInfo, 1220
- setBuffer
 - decaf::io::ByteArrayInputStream, 833
 - decaf::io::ByteArrayOutputStream, 839
- setByte
 - activemq::commands::ActiveMQMapMessage, 281
 - activemq::util::PrimitiveList, 2377
 - activemq::util::PrimitiveMap, 2388
 - activemq::util::PrimitiveValueNode, 2409
 - cms::MapMessage, 1988
- setByteArray
 - activemq::util::PrimitiveList, 2378
 - activemq::util::PrimitiveMap, 2388
 - activemq::util::PrimitiveValueNode, 2409
 - decaf::io::BlockingByteArrayInputStream, 671
 - decaf::io::ByteArrayInputStream, 833
- setByteProperty
 - activemq::commands::ActiveMQMessageTemplate, 339
 - activemq::wireformat::openwire::utils::MessagePropertyInterceptor, 2200
 - cms::Message, 2048
- setBytes
 - activemq::commands::ActiveMQMapMessage, 281
 - cms::MapMessage, 1989
- setCacheEnabled
 - activemq::commands::WireFormatInfo, 3159
 - activemq::wireformat::openwire::OpenWireFormat, 2305
- setCacheSize
 - activemq::commands::WireFormatInfo, 3160
 - activemq::wireformat::openwire::OpenWireFormat, 2305
- setCause
 - activemq::commands::BrokerError, 688
- setChar
 - activemq::commands::ActiveMQMapMessage, 282
 - activemq::util::PrimitiveList, 2378
 - activemq::util::PrimitiveMap, 2388
 - activemq::util::PrimitiveValueNode, 2409
 - cms::MapMessage, 1989
- setClientId
 - activemq::commands::ConnectionInfo, 1133
 - activemq::commands::JournalTopicAck, 1745
 - activemq::commands::RemoveSubscriptionInfo, 2563
 - activemq::commands::SubscriptionInfo, 2910
 - activemq::core::ActiveMQConnectionSupport, 225
- setClientMaster
 - activemq::commands::ConnectionInfo, 1133
- setClose
 - activemq::commands::ConnectionControl, 1058
 - activemq::commands::ConsumerControl, 1169
- setClosed
 - activemq::transport::failover::BackupTransport, 593
- setCloseTimeout
 - activemq::core::ActiveMQConnectionSupport, 225
- setCluster
 - activemq::commands::Message, 2032
- setCMSCorrelationID
 - activemq::commands::ActiveMQMessageTemplate, 339
 - cms::Message, 2049
- setCMSDeliveryMode
 - activemq::commands::ActiveMQMessageTemplate, 339
 - cms::Message, 2049
- setCMSDestination
 - activemq::commands::ActiveMQMessageTemplate, 340
 - cms::Message, 2050
- setCMSExpiration
 - activemq::commands::ActiveMQMessageTemplate, 340
 - cms::Message, 2050
- setCMSMessageID
 - activemq::commands::ActiveMQMessageTemplate, 340
 - cms::Message, 2050
- setCMSPriority
 - activemq::commands::ActiveMQMessageTemplate, 340
 - cms::Message, 2050

- activemq::commands::ActiveMQMessageTemplate
 - 340
- cms::Message, 2051
- set CMSRedelivered
 - activemq::commands::ActiveMQMessageTemplate, 341
 - cms::Message, 2051
- set CMSReplyTo
 - activemq::commands::ActiveMQMessageTemplate, 341
 - cms::Message, 2051
- set CMSTimestamp
 - activemq::commands::ActiveMQMessageTemplate, 341
 - cms::Message, 2052
- set CMSType
 - activemq::commands::ActiveMQMessageTemplate, 342
 - cms::Message, 2052
- set Command
 - activemq::commands::ControlCommand, 1245
 - activemq::wireformat::stomp::StompFrame, 2878
- set CommandId
 - activemq::commands::BaseCommand, 602
 - activemq::commands::Command, 994
 - activemq::commands::PartialCommand, 2321
- set Components
 - activemq::util::CompositeData, 1015
- set Compressed
 - activemq::commands::Message, 2032
- set Connection
 - activemq::commands::ActiveMQTempDestination, 458
- set ConnectionFactory
 - activemq::cmsutil::CmsAccessor, 957
- set ConnectionId
 - activemq::commands::BrokerInfo, 719
 - activemq::commands::ConnectionError, 1082
 - activemq::commands::ConnectionInfo, 1133
 - activemq::commands::ConsumerId, 1194
 - activemq::commands::DestinationInfo, 1394
 - activemq::commands::LocalTransactionId, 1877
 - activemq::commands::ProducerId, 2449
 - activemq::commands::RemoveSubscriptionInfo, 2563
 - activemq::commands::SessionId, 2681
- activemq::commands::TransactionInfo, 3040
- set ConsumerId
 - activemq::commands::ConsumerControl, 1169
 - activemq::commands::ConsumerInfo, 1220
 - activemq::commands::MessageAck, 2059
 - activemq::commands::MessageDispatch, 2087
 - activemq::commands::MessageDispatchNotification, 2119
 - activemq::commands::MessagePull, 2206
- setContent
 - activemq::commands::Message, 2032
- set CorrelationId
 - activemq::commands::Message, 2032
 - activemq::commands::MessagePull, 2206
 - activemq::commands::Response, 2614
- setData
 - activemq::commands::DataArrayResponse, 1270
 - activemq::commands::DataResponse, 1309
 - activemq::commands::PartialCommand, 2321
- setDataStructure
 - activemq::commands::Message, 2032
- setDefaultDestination
 - activemq::cmsutil::CmsTemplate, 979
- setDefaultDestinationName
 - activemq::cmsutil::CmsTemplate, 979
- set DeletedByBroker
 - activemq::commands::ActiveMQBlobMessage, 145
- set DeliveryMode
 - activemq::cmsutil::CachedProducer, 896
 - activemq::cmsutil::CmsTemplate, 979
 - activemq::core::ActiveMQProducer, 373
 - cms::MessageProducer, 2194
- set DeliveryPersistent
 - activemq::cmsutil::CmsTemplate, 979
- set DeliverySequenceId
 - activemq::commands::MessageDispatchNotification, 2119
- set Destination
 - activemq::commands::ConsumerInfo, 1220
 - activemq::commands::DestinationInfo, 1394
 - activemq::commands::JournalQueueAck, 1721
 - activemq::commands::JournalTopicAck, 1745
 - activemq::commands::Message, 2032
 - activemq::commands::MessageAck, 2059

- activemq::commands::MessageDispatch, 2087
- activemq::commands::MessageDispatchNotification, 2119
- activemq::commands::MessagePull, 2206
- activemq::commands::ProducerInfo, 2473
- activemq::commands::SubscriptionInfo, 2910
- setDestinationResolver
 - activemq::cmsutil::CmsDestinationAccessor, 960
- setDisableMessageID
 - activemq::cmsutil::CachedProducer, 896
 - activemq::core::ActiveMQProducer, 373
 - cms::MessageProducer, 2195
- setDisableMessageTimeStamp
 - activemq::cmsutil::CachedProducer, 896
 - activemq::core::ActiveMQProducer, 373
 - cms::MessageProducer, 2195
- setDispatchAsync
 - activemq::commands::ConsumerInfo, 1220
 - activemq::commands::ProducerInfo, 2473
- setDouble
 - activemq::commands::ActiveMQMapMessage, 282
 - activemq::util::PrimitiveList, 2378
 - activemq::util::PrimitiveMap, 2389
 - activemq::util::PrimitiveValueNode, 2409
 - cms::MapMessage, 1989
- setDoubleProperty
 - activemq::commands::ActiveMQMessageTemplate, 342
 - activemq::wireformat::openwire::utils::MessagePropertyInterceptor, 2201
 - cms::Message, 2053
- setDroppable
 - activemq::commands::Message, 2032
- setDuplexConnection
 - activemq::commands::BrokerInfo, 719
- setEnabled
 - activemq::transport::failover::BackupTransportPool, 596
- setenv
 - decaf::lang::System, 2955
- setException
 - activemq::commands::ConnectionError, 1082
 - activemq::commands::ExceptionResponse, 1486
- setExceptionClass
 - activemq::commands::BrokerError, 689
- setExceptionListener
 - activemq::core::ActiveMQConnection, 210
 - cms::Connection, 1055
- setExclusive
 - activemq::commands::ActiveMQDestination, 247
 - activemq::commands::ConsumerInfo, 1220
- setExit
 - activemq::commands::ConnectionControl, 1058
- setExpiration
 - activemq::commands::Message, 2032
- setExplicitQosEnabled
 - activemq::cmsutil::CmsTemplate, 980
- setFailOnClose
 - activemq::transport::mock::MockTransport, 2234
- setFailOnKeepAliveSends
 - activemq::transport::mock::MockTransport, 2235
- setFailOnReceiveMessage
 - activemq::transport::mock::MockTransport, 2235
- setFailOnSendMessage
 - activemq::transport::mock::MockTransport, 2235
- setFailOnStart
 - activemq::transport::mock::MockTransport, 2235
- setFailOnStop
 - activemq::transport::mock::MockTransport, 2235
- setFaultTolerant
 - activemq::commands::ConnectionControl, 1058
- setFaultTolerantConfiguration
 - activemq::commands::BrokerInfo, 719
- setFilter
 - decaf::util::logging::Handler, 1607
 - decaf::util::logging::Logger, 1915
 - decaf::util::logging::StreamHandler, 2891
- setFirstMessageId
 - activemq::commands::MessageAck, 2059
- setFirstNakNumber
 - activemq::commands::ReplayCommand, 2586
- setFloat
 - activemq::commands::ActiveMQMapMessage, 282
 - activemq::util::PrimitiveList, 2379
 - activemq::util::PrimitiveMap, 2389
 - activemq::util::PrimitiveValueNode, 2409
 - cms::MapMessage, 1990
- setFloatProperty
 - activemq::commands::ActiveMQMessageTemplate, 342

- activemq::wireformat::openwire::utils::MessagePropertyInfo, 2201
- cms::Message, 2053
- setFlush
 - activemq::commands::ConsumerControl, 1169
- setFormatId
 - activemq::commands::XATransactionId, 3196
- setFormatter
 - decaf::util::logging::Handler, 1607
 - decaf::util::logging::StreamHandler, 2891
- setFragment
 - activemq::util::CompositeData, 1015
 - decaf::internal::net::URIType, 3131
- setGlobalTransactionId
 - activemq::commands::XATransactionId, 3196
- setGroupID
 - activemq::commands::Message, 2032
- setGroupSequence
 - activemq::commands::Message, 2032
- setHost
 - activemq::util::CompositeData, 1015
 - decaf::internal::net::URIType, 3131
- setInitialDelayTime
 - activemq::transport::inactivity::InactivityMonitor, 1627
- setInitialized
 - activemq::transport::failover::FailoverTransport, 1521
- setInitialReconnectDelay
 - activemq::transport::failover::FailoverTransport, 1521
- setInputStream
 - activemq::transport::IOTransport, 1714
 - decaf::io::Reader, 2519
- setInt
 - activemq::commands::ActiveMQMapMessage, 283
 - activemq::util::PrimitiveList, 2379
 - activemq::util::PrimitiveMap, 2389
 - activemq::util::PrimitiveValueNode, 2410
 - cms::MapMessage, 1990
- setIntProperty
 - activemq::commands::ActiveMQMessageTemplate, 343
 - activemq::wireformat::openwire::utils::MessagePropertyInfo, 2201
 - cms::Message, 2053
- setKeepAlive
 - decaf::net::BufferedSocket, 760
 - decaf::net::Socket, 2790
 - decaf::net::TcpSocket, 2965
- setKeepAliveResponseRequired
 - activemq::transport::inactivity::InactivityMonitor, 1627
- setLastDeliveredSequenceId
 - activemq::commands::RemoveInfo, 2539
 - activemq::core::ActiveMQConsumer, 237
 - activemq::core::ActiveMQSession, 417
- setLastMessageId
 - activemq::commands::MessageAck, 2059
- setLastNakNumber
 - activemq::commands::ReplayCommand, 2586
- setLevel
 - decaf::util::logging::Handler, 1607
 - decaf::util::logging::Logger, 1915
 - decaf::util::logging::LogRecord, 1930
 - decaf::util::logging::StreamHandler, 2891
- setLimit
 - activemq::util::MemoryUsage, 2017
- setList
 - activemq::util::PrimitiveValueNode, 2410
- setLoggerName
 - decaf::util::logging::LogRecord, 1930
- setLong
 - activemq::commands::ActiveMQMapMessage, 283
 - activemq::util::PrimitiveList, 2379
 - activemq::util::PrimitiveMap, 2389
 - activemq::util::PrimitiveValueNode, 2410
 - cms::MapMessage, 1990
- setLongProperty
 - activemq::commands::ActiveMQMessageTemplate, 343
 - activemq::wireformat::openwire::utils::MessagePropertyInfo, 2201
 - cms::Message, 2053
- setMagic
 - activemq::commands::WireFormatInfo, 3160
- setManageable
 - activemq::commands::ConnectionInfo, 1133
- setMap
 - activemq::util::PrimitiveValueNode, 2410
- setMark
 - cms::CMSException, 963
 - decaf::lang::Exception, 1481
 - decaf::lang::Throwable, 2986
- setMarshaledForm
 - activemq::commands::BaseDataStructure, 662
 - activemq::wireformat::MarshalAware, 1995
- setMarshaledProperties
 - activemq::commands::Message, 2032

- activemq::commands::WireFormatInfo, 3160
- setMasterBroker
 - activemq::commands::BrokerInfo, 719
- setMaxCacheSize
 - activemq::state::ConnectionStateTracker, 1165
 - activemq::transport::failover::FailoverTransport, 1521
- setMaximumPendingMessageLimit
 - activemq::commands::ConsumerInfo, 1220
- setMaxInactivityDuration
 - activemq::commands::WireFormatInfo, 3160
 - activemq::wireformat::openwire::OpenWireFormat, 2305
- setMaxInactivityDurationInitialDelay
 - activemq::commands::WireFormatInfo, 3160
- setMaxInactivityDurationInitialDelay
 - activemq::wireformat::openwire::OpenWireFormat, 2305
- setMaxReconnectAttempts
 - activemq::transport::failover::FailoverTransport, 1521
- setMaxReconnectDelay
 - activemq::transport::failover::FailoverTransport, 1521
- setMaxThreads
 - decaf::util::concurrent::ThreadPool, 2982
- setMessage
 - activemq::commands::BrokerError, 689
 - activemq::commands::JournalTrace, 1769
 - activemq::commands::MessageDispatch, 2087
 - decaf::lang::Exception, 1482
 - decaf::util::logging::LogRecord, 1931
- setMessageAck
 - activemq::commands::JournalQueueAck, 1721
- setMessageCount
 - activemq::commands::MessageAck, 2059
- setMessageId
 - activemq::commands::JournalTopicAck, 1745
 - activemq::commands::Message, 2032
 - activemq::commands::MessageDispatchNotification, 2119
 - activemq::commands::MessagePull, 2206
- setMessageIdEnabled
 - activemq::cmsutil::CmsTemplate, 980
- setMessageListener
 - activemq::cmsutil::CachedConsumer, 890
 - activemq::core::ActiveMQConsumer, 237
 - cms::MessageConsumer, 2082
- setMessageSequenceId
 - activemq::commands::JournalTopicAck, 1745
- setMessageTimestampEnabled
 - activemq::cmsutil::CmsTemplate, 980
- setMimeType
 - activemq::commands::ActiveMQBlobMessage, 145
- setName
 - activemq::commands::ActiveMQBlobMessage, 145
- setNetworkBrokerId
 - activemq::commands::NetworkBridgeFilter, 2249
- setNetworkConnection
 - activemq::commands::BrokerInfo, 719
- setNetworkConsumerPath
 - activemq::commands::ConsumerInfo, 1220
- setNetworkProperties
 - activemq::commands::BrokerInfo, 719
- setNetworkSubscription
 - activemq::commands::ConsumerInfo, 1220
- setNetworkTTL
 - activemq::commands::NetworkBridgeFilter, 2249
- setNoLocal
 - activemq::cmsutil::CmsTemplate, 980
 - activemq::commands::ConsumerInfo, 1220
- setNoRangeAcks
 - activemq::commands::ConsumerInfo, 1220
- setNumReceivedMessageBeforeFail
 - activemq::transport::mock::MockTransport, 2235
- setNumReceivedMessages
 - activemq::transport::mock::MockTransport, 2235
- setNumSentKeepAlives
 - activemq::transport::mock::MockTransport, 2235
- setNumSentKeepAlivesBeforeFail
 - activemq::transport::mock::MockTransport, 2235
- setNumSentMessageBeforeFail
 - activemq::transport::mock::MockTransport, 2235
- setNumSentMessages
 - activemq::transport::mock::MockTransport, 2235
- setObjectId
 - activemq::commands::RemoveInfo, 2539
- setOpaque
 - decaf::internal::net::URIType, 3131
- setOperationType

- activemq::commands::DestinationInfo, 1394
- setOptimizedAcknowledge
 - activemq::commands::ConsumerInfo, 1220
- setOrdered
 - activemq::commands::ActiveMQDestination, 247
- setOrderedTarget
 - activemq::commands::ActiveMQDestination, 247
- setOriginalDestination
 - activemq::commands::Message, 2032
- setOriginalTransactionId
 - activemq::commands::Message, 2032
- setOutgoingListener
 - activemq::transport::mock::MockTransport, 2235
- setOutputStream
 - activemq::transport::IOTransport, 1714
 - decaf::io::Writer, 3187
- setParameters
 - activemq::util::CompositeData, 1015
- setPassword
 - activemq::commands::ConnectionInfo, 1133
 - activemq::core::ActiveMQConnectionFactory, 215
 - activemq::core::ActiveMQConnectionSupport, 225
- setPath
 - activemq::util::CompositeData, 1015
 - decaf::internal::net::URIType, 3131
- setPeerBrokerInfos
 - activemq::commands::BrokerInfo, 719
- setPersistent
 - activemq::commands::Message, 2032
- setPhysicalName
 - activemq::commands::ActiveMQDestination, 248
- setPooledThreadListener
 - decaf::util::concurrent::PooledThread, 2365
- setPort
 - decaf::internal::net::URIType, 3131
- setPreferedWireFormatInfo
 - activemq::wireformat::openwire::OpenWireFormat, 2305
- setPrefetch
 - activemq::commands::ConsumerControl, 1169
- setPrefetchSize
 - activemq::commands::ConsumerInfo, 1220
- setPrepared
 - activemq::state::TransactionState, 3061
- setPreparedResult
 - activemq::state::TransactionState, 3061
- setPriority
 - activemq::cmsutil::CachedProducer, 896
 - activemq::cmsutil::CmsTemplate, 980
 - activemq::commands::ConsumerInfo, 1220
 - activemq::commands::Message, 2032
 - activemq::core::ActiveMQProducer, 374
 - cms::MessageProducer, 2195
- setProducerId
 - activemq::commands::Message, 2032
 - activemq::commands::MessageId, 2145
 - activemq::commands::ProducerAck, 2423
 - activemq::commands::ProducerInfo, 2473
- setProducerSequenceId
 - activemq::commands::MessageId, 2145
- setProducerWindowSize
 - activemq::core::ActiveMQConnectionSupport, 225
- setProperties
 - activemq::commands::WireFormatInfo, 3161
 - activemq::util::ActiveMQProperties, 378
 - decaf::util::logging::LogManager, 1927
- setProperty
 - activemq::util::ActiveMQProperties, 378
 - activemq::wireformat::stomp::StompFrame, 2878
 - cms::CMSProperties, 967
 - decaf::util::Properties, 2501
- setPubSubDomain
 - activemq::cmsutil::CmsDestinationAccessor, 960
 - activemq::cmsutil::CmsTemplate, 980
- setQuery
 - decaf::internal::net::URIType, 3131
- setRandomize
 - activemq::transport::failover::FailoverTransport, 1521
 - activemq::transport::failover::URIPool, 3119
- setReadCheckTime
 - activemq::transport::inactivity::InactivityMonitor, 1627
- setReadOnly
 - decaf::internal::nio::ByteBuffer, 827
 - decaf::internal::nio::CharArrayBuffer, 929
 - decaf::internal::nio::DoubleArrayBuffer, 1459
 - decaf::internal::nio::FloatArrayBuffer, 1561
 - decaf::internal::nio::IntArrayBuffer, 1640
 - decaf::internal::nio::LongArrayBuffer, 1954
 - decaf::internal::nio::ShortArrayBuffer, 2745
- setReadOnlyBody

- activemq::commands::Message, 2032
- setReadOnlyProperties
 - activemq::commands::Message, 2033
- setReceiveBufferSize
 - decaf::net::BufferedSocket, 760
 - decaf::net::Socket, 2790
 - decaf::net::TcpSocket, 2965
- setReceiveTimeout
 - activemq::cmsutil::CmsTemplate, 981
- setRecievedByDFBbridge
 - activemq::commands::Message, 2033
- setReconnectDelay
 - activemq::transport::failover::FailoverTransport, 1521
- setRedeliveryCounter
 - activemq::commands::Message, 2033
 - activemq::commands::MessageDispatch, 2087
- setRemoteBlobUrl
 - activemq::commands::ActiveMQBlobMessage, 145
- setReplyTo
 - activemq::commands::Message, 2033
- setResponse
 - activemq::transport::correlator::FutureResponse, 1601
- setResponseBuilder
 - activemq::transport::mock::InternalCommandSender, 1691
 - activemq::transport::mock::MockTransport, 2235
- setResponseRequired
 - activemq::commands::BaseCommand, 602
 - activemq::commands::Command, 994
- setRestoreConsumers
 - activemq::state::ConnectionStateTracker, 1165
- setRestoreProducers
 - activemq::state::ConnectionStateTracker, 1165
- setRestoreSessions
 - activemq::state::ConnectionStateTracker, 1165
- setRestoreTransaction
 - activemq::state::ConnectionStateTracker, 1165
- setResult
 - activemq::commands::IntegerResponse, 1669
- setResume
 - activemq::commands::ConnectionControl, 1058
- setRetroactive
 - activemq::commands::ConsumerInfo, 1220
- setReuseAddress
 - decaf::net::BufferedSocket, 760
 - decaf::net::Socket, 2791
 - decaf::net::TcpSocket, 2965
- setScheduledTime
 - decaf::util::TimerTask, 3002
- setScheme
 - activemq::util::CompositeData, 1015
 - decaf::internal::net::URIType, 3132
- setSchemeSpecificPart
 - decaf::internal::net::URIType, 3132
- setSeed
 - decaf::util::Random, 2517
- setSelector
 - activemq::commands::ConsumerInfo, 1220
 - activemq::commands::SubscriptionInfo, 2910
- setSendBufferSize
 - decaf::net::BufferedSocket, 760
 - decaf::net::Socket, 2791
 - decaf::net::TcpSocket, 2965
- setSendTimeout
 - activemq::core::ActiveMQConnectionSupport, 226
 - activemq::core::ActiveMQProducer, 374
- setServerAuthority
 - decaf::internal::net::URIType, 3132
- setServerName
 - activemq::commands::DiscoveryEvent, 1419
- setSessionAcknowledgeMode
 - activemq::cmsutil::CmsAccessor, 957
- setSessionId
 - activemq::commands::ConsumerId, 1194
 - activemq::commands::ProducerId, 2449
 - activemq::commands::SessionInfo, 2704
- setShort
 - activemq::commands::ActiveMQMapMessage, 283
 - activemq::util::PrimitiveList, 2380
 - activemq::util::PrimitiveMap, 2389
 - activemq::util::PrimitiveValueNode, 2410
 - cms::MapMessage, 1991
- setShortProperty
 - activemq::commands::ActiveMQMessageTemplate, 343
 - activemq::wireformat::openwire::utils::MessagePropertyIntercept, 2202
 - cms::Message, 2054
- setSize
 - activemq::commands::ProducerAck, 2423
- setSizePrefixDisabled
 - activemq::commands::WireFormatInfo, 3161

- activemq::wireformat::openwire::OpenWireFormat, 2305
- setSlaveBroker
 - activemq::commands::BrokerInfo, 719
- setSoLinger
 - decaf::net::BufferedSocket, 761
 - decaf::net::Socket, 2791
 - decaf::net::TcpSocket, 2966
- setSoTimeout
 - decaf::net::BufferedSocket, 761
 - decaf::net::Socket, 2791
 - decaf::net::TcpSocket, 2966
- setSource
 - decaf::internal::net::URIType, 3132
- setSourceFile
 - decaf::util::logging::LogRecord, 1931
- setSourceFunction
 - decaf::util::logging::LogRecord, 1931
- setSourceLine
 - decaf::util::logging::LogRecord, 1931
- setStackTrace
 - decaf::lang::Exception, 1482
- setStackTraceElements
 - activemq::commands::BrokerError, 689
- setStackTraceEnabled
 - activemq::commands::WireFormatInfo, 3161
 - activemq::wireformat::openwire::OpenWireFormat, 2306
- setStart
 - activemq::commands::ConsumerControl, 1169
- setStop
 - activemq::commands::ConsumerControl, 1169
- setString
 - activemq::commands::ActiveMQMapMessage, 283
 - activemq::util::PrimitiveList, 2380
 - activemq::util::PrimitiveMap, 2390
 - activemq::util::PrimitiveValueNode, 2411
 - cms::MapMessage, 1991
- setStringProperty
 - activemq::commands::ActiveMQMessageTemplate, 344
 - activemq::wireformat::openwire::utils::MessageProperty, 2202
 - cms::Message, 2054
- setSubscriptionName
 - activemq::commands::RemoveSubscriptionInfo, 2563
 - activemq::commands::SubscriptionInfo, 2910
- setSubscribedDestination
 - activemq::commands::SubscriptionInfo, 2910
- setSubscriptionName
 - activemq::commands::ConsumerInfo, 1220
- setSubscriptionName
 - activemq::commands::JournalTopicAck, 1745
- setSuspend
 - activemq::commands::ConnectionControl, 1058
- setSynchronizationRegistered
 - activemq::core::ActiveMQConsumer, 237
- setTargetConsumerId
 - activemq::commands::Message, 2033
- setTcpNoDelay
 - decaf::net::TcpSocket, 2966
- setTcpNoDelayEnabled
 - activemq::commands::WireFormatInfo, 3161
 - activemq::wireformat::openwire::OpenWireFormat, 2306
- setText
 - activemq::commands::ActiveMQTextMessage, 528, 529
 - cms::TextMessage, 2975
- setTightEncodingEnabled
 - activemq::commands::WireFormatInfo, 3161
 - activemq::wireformat::openwire::OpenWireFormat, 2306
- setTime
 - decaf::util::Date, 1380
- setTimeout
 - activemq::commands::DestinationInfo, 1394
 - activemq::commands::MessagePull, 2206
 - activemq::transport::failover::FailoverTransport, 1521
- setTimestamp
 - activemq::commands::Message, 2033
 - decaf::util::logging::LogRecord, 1931
- setTimeToLive
 - activemq::cmsutil::CachedProducer, 897
 - activemq::cmsutil::CmsTemplate, 981
 - activemq::core::ActiveMQProducer, 374
 - activemq::core::ActiveMQProducer, 2196
- setTrackMessages
 - activemq::state::ConnectionStateTracker, 1165
 - activemq::transport::failover::FailoverTransport, 1521
- setTrackTransactions
 - activemq::state::ConnectionStateTracker, 1165

- setTransactionId
 - activemq::commands::JournalTopicAck, 1745
 - activemq::commands::JournalTransaction, 1792
 - activemq::commands::Message, 2033
 - activemq::commands::MessageAck, 2059
 - activemq::commands::TransactionInfo, 3040
- setTransport
 - activemq::transport::failover::BackupTransport, 593
 - activemq::transport::mock::InternalCommandListener, 1691
- setTransportListener
 - activemq::transport::failover::FailoverTransport, 1521
 - activemq::transport::IOTransport, 1714
 - activemq::transport::mock::MockTransport, 2236
 - activemq::transport::Transport, 3070
 - activemq::transport::TransportFilter, 3079
- setTreadId
 - decaf::util::logging::LogRecord, 1931
- setType
 - activemq::commands::JournalTransaction, 1792
 - activemq::commands::Message, 2033
 - activemq::commands::TransactionInfo, 3040
- setUri
 - activemq::transport::failover::BackupTransport, 593
- setUsage
 - activemq::util::MemoryUsage, 2017
- setUseAsyncSend
 - activemq::core::ActiveMQConnectionSupport, 226
- setUseExponentialBackOff
 - activemq::transport::failover::FailoverTransport, 1522
- setUseParentHandlers
 - decaf::util::logging::Logger, 1916
- setUserID
 - activemq::commands::Message, 2033
- setUserInfo
 - decaf::internal::net::URIType, 3132
- setUserName
 - activemq::commands::ConnectionInfo, 1133
- setUsername
 - activemq::core::ActiveMQConnectionFactory, 216
- activemq::core::ActiveMQConnectionSupport, 226
- setValid
 - decaf::internal::net::URIType, 3133
- setValue
 - activemq::commands::BrokerId, 693
 - activemq::commands::ConnectionId, 1108
 - activemq::commands::ConsumerId, 1194
 - activemq::commands::LocalTransactionId, 1877
 - activemq::commands::ProducerId, 2449
 - activemq::commands::SessionId, 2681
 - activemq::util::PrimitiveValueNode, 2411
 - decaf::util::Map::Entry, 1473
- setVersion
 - activemq::commands::WireFormatInfo, 3162
 - activemq::wireformat::openwire::OpenWireFormat, 2306
 - activemq::wireformat::stomp::StompWireFormat, 2885
 - activemq::wireformat::WireFormat, 3150
- setWasPrepared
 - activemq::commands::JournalTransaction, 1792
- setWindowSize
 - activemq::commands::ProducerInfo, 2473
- setWireFormat
 - activemq::transport::failover::FailoverTransport, 1522
 - activemq::transport::IOTransport, 1714
 - activemq::transport::mock::MockTransport, 2236
 - activemq::transport::Transport, 3070
 - activemq::transport::TransportFilter, 3079
- setWriteCheckTime
 - activemq::transport::inactivity::InactivityMonitor, 1627
- Short
 - decaf::lang::Short, 2731
- SHORT_TYPE
 - activemq::util::PrimitiveValueNode, 2402
- ShortArrayBuffer
 - decaf::internal::nio::ShortArrayBuffer, 2740, 2741
- ShortBuffer
 - decaf::nio::ShortBuffer, 2748
- shortValue
 - activemq::util::PrimitiveValueNode::PrimitiveValue, 2396
 - decaf::lang::Byte, 782
 - decaf::lang::Character, 921
 - decaf::lang::Double, 1449
 - decaf::lang::Float, 1552

- decaf::lang::Integer, 1663
- decaf::lang::Long, 1944
- decaf::lang::Number, 2284
- decaf::lang::Short, 2736
- shutdown
 - activemq::state::ConnectionState, 1160
 - activemq::state::SessionState, 2728
 - activemq::state::TransactionState, 3061
 - activemq::threads::CompositeTaskRunner, 1018
 - activemq::threads::DedicatedTaskRunner, 1383
 - activemq::threads::TaskRunner, 2958
- ShutdownInfo
 - activemq::commands::ShutdownInfo, 2758
- ShutdownInfoMarshaller
 - activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller, 2761
 - activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller, 2773
 - activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller, 2769
 - activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller, 2765
 - activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller, 2777
- shutdownLibrary
 - activemq::library::ActiveMQCPP, 239
- shutdownRuntime
 - decaf::lang::Runtime, 2644
- shutdownTransport
 - activemq::core::ActiveMQConnectionSupport, 226
- signal
 - decaf::util::concurrent::locks::Condition, 1046
- signalAll
 - decaf::util::concurrent::locks::Condition, 1046
- SignatureException
 - decaf::security::SignatureException, 2780, 2781
- signum
 - decaf::lang::Integer, 1663
 - decaf::lang::Long, 1944
 - decaf::lang::Math, 2009, 2010
- SimpleFormatter
 - decaf::util::logging::SimpleFormatter, 2783
- SimpleLogger
 - decaf::util::logging::SimpleLogger, 2785
- SIZE
 - decaf::lang::Byte, 784
 - decaf::lang::Character, 921
 - decaf::lang::Double, 1452
 - decaf::lang::Float, 1554
 - decaf::lang::Integer, 1666
 - decaf::lang::Long, 1947
 - decaf::lang::Short, 2738
- size
 - activemq::commands::ProducerAck, 2423
 - activemq::core::MessageDispatchChannel, 2093
 - activemq::wireformat::openwire::utils::HexTable, 1610
 - decaf::internal::util::TimerTaskHeap, 3005
 - decaf::io::ByteArrayOutputStream, 839
 - decaf::io::DataOutputStream, 1302
 - decaf::util::Collection, 990
 - decaf::util::concurrent::ConcurrentStlMap, 1037
 - decaf::util::concurrent::ConcurrentStlQueue, 2952
 - decaf::util::PriorityQueue, 2419
 - decaf::util::StlList, 2844
 - decaf::util::StlQueue, 2863
 - decaf::util::StlSet, 2871
- skip
 - decaf::internal::io::StandardInputStream, 2822
 - decaf::io::BlockingByteArrayInputStream, 671
 - decaf::io::BufferedInputStream, 751
 - decaf::io::ByteArrayInputStream, 833
 - decaf::io::DataInputStream, 1299
 - decaf::io::FilterInputStream, 1534
 - decaf::io::InputStream, 1633
 - decaf::net::SocketInputStream, 2801
- slaveBroker
 - activemq::commands::BrokerInfo, 721
- sleep
 - decaf::util::concurrent::TimeUnit, 3009
- slice
 - decaf::internal::nio::ByteBuffer, 827
 - decaf::internal::nio::CharArrayBuffer, 929
 - decaf::internal::nio::DoubleArrayBuffer, 1459
 - decaf::internal::nio::FloatArrayBuffer, 1561
 - decaf::internal::nio::IntArrayBuffer, 1641
 - decaf::internal::nio::LongArrayBuffer, 1955
 - decaf::internal::nio::ShortArrayBuffer, 2745
 - decaf::nio::ByteBuffer, 872
 - decaf::nio::CharBuffer, 944
 - decaf::nio::DoubleBuffer, 1469
 - decaf::nio::FloatBuffer, 1571

decaf::nio::IntBuffer, 1651	src/main/activemq/commands/ActiveMQMessage.h,
decaf::nio::LongBuffer, 1965	3228
decaf::nio::ShortBuffer, 2755	src/main/activemq/commands/ActiveMQMessageTemplate.h,
SocketAddress	3228
decaf::net::ServerSocket, 2662	src/main/activemq/commands/ActiveMQObjectMessage.h,
decaf::net::Socket, 2787	3229
SocketException	src/main/activemq/commands/ActiveMQQueue.h,
decaf::net::SocketException, 2793, 2794	3229
SocketHandle	src/main/activemq/commands/ActiveMQStreamMessage.h,
decaf::net::ServerSocket, 2662	3230
decaf::net::Socket, 2787	src/main/activemq/commands/ActiveMQTempDestination.h,
SocketInputStream	3231
decaf::net::SocketInputStream, 2798	src/main/activemq/commands/ActiveMQTempQueue.h,
SocketOutputStream	3231
decaf::net::SocketOutputStream, 2805	src/main/activemq/commands/ActiveMQTempTopic.h,
SocketTimeoutException	3232
decaf::net::SocketTimeoutException, 2810,	src/main/activemq/commands/ActiveMQTextMessage.h,
2811	3232
sqrt	src/main/activemq/commands/ActiveMQTopic.h,
decaf::lang::Math, 2011	3233
src/main/activemq/cmsutil/CachedConsumer.h,	src/main/activemq/commands/BaseCommand.h,
3219	3233
src/main/activemq/cmsutil/CachedProducer.h,	src/main/activemq/commands/BaseDataStructure.h,
3219	3234
src/main/activemq/cmsutil/CmsAccessor.h,	src/main/activemq/commands/BooleanExpression.h,
3220	3234
src/main/activemq/cmsutil/CmsDestinationAccess	src/main/activemq/commands/BrokerError.h,
3220	3235
src/main/activemq/cmsutil/CmsTemplate.h,	src/main/activemq/commands/BrokerId.h,
3221	3235
src/main/activemq/cmsutil/DestinationResolver.h,	src/main/activemq/commands/BrokerInfo.h,
3221	3236
src/main/activemq/cmsutil/DynamicDestinationReso	src/main/activemq/commands/Command.h,
3222	3236
src/main/activemq/cmsutil/MessageCreator.h,	src/main/activemq/commands/ConnectionControl.h,
3222	3237
src/main/activemq/cmsutil/PooledSession.h,	src/main/activemq/commands/ConnectionError.h,
3223	3237
src/main/activemq/cmsutil/ProducerCallback.h,	src/main/activemq/commands/ConnectionId.h,
3223	3238
src/main/activemq/cmsutil/ResourceLifecycleMan	src/main/activemq/commands/ConnectionInfo.h,
3224	3238
src/main/activemq/cmsutil/SessionCallback.h,	src/main/activemq/commands/ConsumerControl.h,
3225	3239
src/main/activemq/cmsutil/SessionPool.h,	src/main/activemq/commands/ConsumerId.h,
3225	3239
src/main/activemq/commands/ActiveMQBlobMessage	src/main/activemq/commands/ConsumerInfo.h,
3226	3240
src/main/activemq/commands/ActiveMQBytesMessage	src/main/activemq/commands/ControlCommand.h,
3226	3241
src/main/activemq/commands/ActiveMQDestination	src/main/activemq/commands/DataArrayResponse.h,
3227	3241
src/main/activemq/commands/ActiveMQMapMessage	src/main/activemq/commands/DataResponse.h,
3227	3242

src/main/activemq/commands/DataStructure.h, src/main/activemq/commands/Response.h,
 3242 3257
 src/main/activemq/commands/DestinationInfo.h,src/main/activemq/commands/SessionId.h,
 3242 3257
 src/main/activemq/commands/DiscoveryEvent.h,src/main/activemq/commands/SessionInfo.h,
 3243 3258
 src/main/activemq/commands/ExceptionResponse.h,src/main/activemq/commands/ShutdownInfo.h,
 3244 3259
 src/main/activemq/commands/FlushCommand.hsrc/main/activemq/commands/SubscriptionInfo.h,
 3244 3259
 src/main/activemq/commands/IntegerResponse.hsrc/main/activemq/commands/TransactionId.h,
 3245 3260
 src/main/activemq/commands/JournalQueueAck.hsrc/main/activemq/commands/TransactionInfo.h,
 3245 3260
 src/main/activemq/commands/JournalTopicAck.hsrc/main/activemq/commands/WireFormatInfo.h,
 3246 3261
 src/main/activemq/commands/JournalTrace.h, src/main/activemq/commands/XATransactionId.h,
 3246 3261
 src/main/activemq/commands/JournalTransactionId.h,src/main/activemq/core/ActiveMQAckHandler.h,
 3247 3262
 src/main/activemq/commands/KeepAliveInfo.h, src/main/activemq/core/ActiveMQConnection.h,
 3247 3262
 src/main/activemq/commands/LastPartialCommand.h,src/main/activemq/core/ActiveMQConnectionFactory.h,
 3248 3263
 src/main/activemq/commands/LocalTransactionId.h,src/main/activemq/core/ActiveMQConnectionMetaData.h,
 3248 3264
 src/main/activemq/commands/Message.h, src/main/activemq/core/ActiveMQConnectionSupport.h,
 3249 3264
 src/main/activemq/commands/MessageAck.h, src/main/activemq/core/ActiveMQConstants.h,
 3250 3265
 src/main/activemq/commands/MessageDispatch.h,src/main/activemq/core/ActiveMQConsumer.h,
 3251 3265
 src/main/activemq/commands/MessageDispatchNotification.h,src/main/activemq/core/ActiveMQProducer.h,
 3251 3266
 src/main/activemq/commands/MessageId.h, src/main/activemq/core/ActiveMQSession.h,
 3252 3267
 src/main/activemq/commands/MessagePull.h, src/main/activemq/core/ActiveMQSessionExecutor.h,
 3252 3268
 src/main/activemq/commands/NetworkBridgeFilter.h,src/main/activemq/core/ActiveMQTransactionContext.h,
 3253 3268
 src/main/activemq/commands/PartialCommand.h,src/main/activemq/core/DispatchData.h, 3269
 3253 src/main/activemq/core/Dispatcher.h, 3269
 src/main/activemq/commands/ProducerAck.h, src/main/activemq/core/MessageDispatchChannel.h,
 3254 3270
 src/main/activemq/commands/ProducerId.h, src/main/activemq/core/Synchronization.h,
 3254 3270
 src/main/activemq/commands/ProducerInfo.h, src/main/activemq/exceptions/ActiveMQException.h,
 3255 3271
 src/main/activemq/commands/RemoveInfo.h, src/main/activemq/exceptions/BrokerException.h,
 3255 3271
 src/main/activemq/commands/RemoveSubscriptionId.h,src/main/activemq/exceptions/ExceptionDefines.h,
 3256 3272
 src/main/activemq/commands/ReplayCommand.h,src/main/activemq/io/LoggingInputStream.h,
 3256 3276

src/main/activemq/io/LoggingOutputStream.h, 3293
 3277
 src/main/activemq/library/ActiveMQCPP.h, 3294
 3277
 src/main/activemq/state/CommandVisitor.h, 3294
 3278
 src/main/activemq/state/CommandVisitorAdapter.h, 3295
 3278
 src/main/activemq/state/ConnectionState.h, 3295
 3279
 src/main/activemq/state/ConnectionStateTracker.h, 3296
 3280
 src/main/activemq/state/ConsumerState.h, 3297
 3281
 src/main/activemq/state/ProducerState.h, 3297
 3281
 src/main/activemq/state/SessionState.h, 3282
 src/main/activemq/state/Tracked.h, 3282
 src/main/activemq/state/TransactionState.h, 3298
 3283
 src/main/activemq/threads/CompositeTask.h, 3299
 3284
 src/main/activemq/threads/CompositeTaskRunner.h, 3300
 3284
 src/main/activemq/threads/DedicatedTaskRunner.h, 3300
 3285
 src/main/activemq/threads/Task.h, 3285
 src/main/activemq/threads/TaskRunner.h, 3301
 3286
 src/main/activemq/transport/AbstractTransportFactory.h, 3302
 3286
 src/main/activemq/transport/CompositeTransport.h, 3302
 3287
 src/main/activemq/transport/correlator/FutureResponse.h, 3303
 3287
 src/main/activemq/transport/correlator/ResponseCorrelator.h, 3303
 3288
 src/main/activemq/transport/DefaultTransportListener.h, 3305
 3288
 src/main/activemq/transport/failover/BackupTransport.h, 3306
 3289
 src/main/activemq/transport/failover/BackupTransportPool.h, 3307
 3289
 src/main/activemq/transport/failover/CloseTransportTask.h, 3308
 3290
 src/main/activemq/transport/failover/FailoverTransport.h, 3309
 3290
 src/main/activemq/transport/failover/FailoverTransportFactory.h, 3309
 3291
 src/main/activemq/transport/failover/FailoverTransportListener.h, 3310
 3292
 src/main/activemq/transport/failover/URIPool.h, 3310
 3292
 src/main/activemq/transport/inactivity/InactivityMonitor, 3311

src/main/activemq/wireformat/openwire/marshaller/DataResponseWireFormat/openwire/marshaller/v1/DataResponseWireFormat 3312 3396

src/main/activemq/wireformat/openwire/marshaller/PrimitiveTypeMarshaller/openwire/marshaller/v1/DestinationWireFormat 3312 3400

src/main/activemq/wireformat/openwire/marshaller/ActiveMQBlobMessageMarshaller/openwire/marshaller/v1/DiscoveryExchangeWireFormat 3313 3403

src/main/activemq/wireformat/openwire/marshaller/ActiveMQBytesMessageMarshaller/openwire/marshaller/v1/ExceptionResponseWireFormat 3316 3406

src/main/activemq/wireformat/openwire/marshaller/ActiveMQDestinationWireFormat/openwire/marshaller/v1/FlushCommandWireFormat 3320 3410

src/main/activemq/wireformat/openwire/marshaller/ActiveMQMapMessageMarshaller/openwire/marshaller/v1/IntegerResponseWireFormat 3323 3413

src/main/activemq/wireformat/openwire/marshaller/ActiveMQMessageMarshaller/openwire/marshaller/v1/JournalQueueWireFormat 3326 3416

src/main/activemq/wireformat/openwire/marshaller/ActiveMQObjectMessageMarshaller/openwire/marshaller/v1/JournalTopicWireFormat 3330 3420

src/main/activemq/wireformat/openwire/marshaller/ActiveMQQueueWireFormat/openwire/marshaller/v1/JournalTraceWireFormat 3333 3423

src/main/activemq/wireformat/openwire/marshaller/ActiveMQSimpleMessageMarshaller/openwire/marshaller/v1/JournalTransactionWireFormat 3336 3426

src/main/activemq/wireformat/openwire/marshaller/ActiveMQSyncDestinationMarshaller/openwire/marshaller/v1/KeepAliveResponseWireFormat 3340 3430

src/main/activemq/wireformat/openwire/marshaller/ActiveMQSyncQueueMarshaller/openwire/marshaller/v1/LastPartialMessageWireFormat 3343 3433

src/main/activemq/wireformat/openwire/marshaller/ActiveMQSyncQueueMarshaller/openwire/marshaller/v1/LocalTransactionWireFormat 3346 3436

src/main/activemq/wireformat/openwire/marshaller/ActiveMQTextMessageMarshaller/openwire/marshaller/v1/MarshallerFactory 3350 3440

src/main/activemq/wireformat/openwire/marshaller/ActiveMQTopicWireFormat/openwire/marshaller/v1/MessageAcknowledgeWireFormat 3353 3442

src/main/activemq/wireformat/openwire/marshaller/ActiveMQCommandMarshaller/openwire/marshaller/v1/MessageDispatchWireFormat 3356 3445

src/main/activemq/wireformat/openwire/marshaller/ActiveMQBrokerMarshaller/openwire/marshaller/v1/MessageDispatchWireFormat 3360 3449

src/main/activemq/wireformat/openwire/marshaller/ActiveMQBrokerMarshaller/openwire/marshaller/v1/MessageIdWireFormat 3363 3452

src/main/activemq/wireformat/openwire/marshaller/ActiveMQConnectiveControlMarshaller/openwire/marshaller/v1/MessageMarshaller 3366 3456

src/main/activemq/wireformat/openwire/marshaller/ActiveMQConnectiveControlMarshaller/openwire/marshaller/v1/MessagePublisher 3370 3459

src/main/activemq/wireformat/openwire/marshaller/ActiveMQConnectiveControlMarshaller/openwire/marshaller/v1/NetworkBridge 3373 3462

src/main/activemq/wireformat/openwire/marshaller/ActiveMQConnectiveControlMarshaller/openwire/marshaller/v1/PartialCommandWireFormat 3376 3466

src/main/activemq/wireformat/openwire/marshaller/ActiveMQConnectiveControlMarshaller/openwire/marshaller/v1/ProducerAcknowledgeWireFormat 3380 3469

src/main/activemq/wireformat/openwire/marshaller/ActiveMQConnectiveControlMarshaller/openwire/marshaller/v1/ProducerIdWireFormat 3383 3472

src/main/activemq/wireformat/openwire/marshaller/ActiveMQConnectiveControlMarshaller/openwire/marshaller/v1/ProducerInfoWireFormat 3386 3476

src/main/activemq/wireformat/openwire/marshaller/ActiveMQConnectiveControlMarshaller/openwire/marshaller/v1/RemoveInfoWireFormat 3390 3479

src/main/activemq/wireformat/openwire/marshaller/ActiveMQConnectiveControlMarshaller/openwire/marshaller/v1/RemoveSubscriptionWireFormat 3393 3482

src/main/activemq/wireformat/openwire/marshaller/v1/ReplaceCommandMarshaller/openwire/marshaller/v2/Connection	3486	3370
src/main/activemq/wireformat/openwire/marshaller/v1/ResponsiveMarshaller/wireformat/openwire/marshaller/v2/Connection	3489	3374
src/main/activemq/wireformat/openwire/marshaller/v1/SessionIdMarshaller/wireformat/openwire/marshaller/v2/Connection	3492	3377
src/main/activemq/wireformat/openwire/marshaller/v1/SessionInfoMarshaller/wireformat/openwire/marshaller/v2/ConsumerC	3496	3380
src/main/activemq/wireformat/openwire/marshaller/v1/SessionInfoMarshaller/wireformat/openwire/marshaller/v2/ConsumerC	3499	3384
src/main/activemq/wireformat/openwire/marshaller/v1/SessionInfoMarshaller/wireformat/openwire/marshaller/v2/ConsumerC	3502	3387
src/main/activemq/wireformat/openwire/marshaller/v1/SessionInfoMarshaller/wireformat/openwire/marshaller/v2/ConsumerC	3506	3390
src/main/activemq/wireformat/openwire/marshaller/v1/SessionInfoMarshaller/wireformat/openwire/marshaller/v2/ConsumerC	3509	3394
src/main/activemq/wireformat/openwire/marshaller/v1/SessionInfoMarshaller/wireformat/openwire/marshaller/v2/ConsumerC	3512	3397
src/main/activemq/wireformat/openwire/marshaller/v1/SessionInfoMarshaller/wireformat/openwire/marshaller/v2/ConsumerC	3516	3400
src/main/activemq/wireformat/openwire/marshaller/v1/SessionInfoMarshaller/wireformat/openwire/marshaller/v2/ConsumerC	3314	3404
src/main/activemq/wireformat/openwire/marshaller/v1/SessionInfoMarshaller/wireformat/openwire/marshaller/v2/ConsumerC	3317	3407
src/main/activemq/wireformat/openwire/marshaller/v1/SessionInfoMarshaller/wireformat/openwire/marshaller/v2/ConsumerC	3320	3410
src/main/activemq/wireformat/openwire/marshaller/v1/SessionInfoMarshaller/wireformat/openwire/marshaller/v2/ConsumerC	3324	3414
src/main/activemq/wireformat/openwire/marshaller/v1/SessionInfoMarshaller/wireformat/openwire/marshaller/v2/ConsumerC	3327	3417
src/main/activemq/wireformat/openwire/marshaller/v1/SessionInfoMarshaller/wireformat/openwire/marshaller/v2/ConsumerC	3330	3420
src/main/activemq/wireformat/openwire/marshaller/v1/SessionInfoMarshaller/wireformat/openwire/marshaller/v2/ConsumerC	3334	3424
src/main/activemq/wireformat/openwire/marshaller/v1/SessionInfoMarshaller/wireformat/openwire/marshaller/v2/ConsumerC	3337	3427
src/main/activemq/wireformat/openwire/marshaller/v1/SessionInfoMarshaller/wireformat/openwire/marshaller/v2/ConsumerC	3340	3430
src/main/activemq/wireformat/openwire/marshaller/v1/SessionInfoMarshaller/wireformat/openwire/marshaller/v2/ConsumerC	3344	3434
src/main/activemq/wireformat/openwire/marshaller/v1/SessionInfoMarshaller/wireformat/openwire/marshaller/v2/ConsumerC	3347	3437
src/main/activemq/wireformat/openwire/marshaller/v1/SessionInfoMarshaller/wireformat/openwire/marshaller/v2/ConsumerC	3350	3440
src/main/activemq/wireformat/openwire/marshaller/v1/SessionInfoMarshaller/wireformat/openwire/marshaller/v2/ConsumerC	3354	3443
src/main/activemq/wireformat/openwire/marshaller/v1/SessionInfoMarshaller/wireformat/openwire/marshaller/v2/ConsumerC	3357	3446
src/main/activemq/wireformat/openwire/marshaller/v1/SessionInfoMarshaller/wireformat/openwire/marshaller/v2/ConsumerC	3360	3450
src/main/activemq/wireformat/openwire/marshaller/v1/SessionInfoMarshaller/wireformat/openwire/marshaller/v2/ConsumerC	3364	3453
src/main/activemq/wireformat/openwire/marshaller/v1/SessionInfoMarshaller/wireformat/openwire/marshaller/v2/ConsumerC	3367	3456

src/main/activemq/wireformat/openwire/marshalsrv2/MessagePullMarshaller/openwire/marshall/v3/ActiveMQT
3460 3344

src/main/activemq/wireformat/openwire/marshalsrv2/NetworkBridgeFilterMarshaller/openwire/marshall/v3/ActiveMQT
3463 3348

src/main/activemq/wireformat/openwire/marshalsrv2/PartialCommandMarshaller/openwire/marshall/v3/ActiveMQT
3466 3351

src/main/activemq/wireformat/openwire/marshalsrv2/ProducerAckMarshaller/openwire/marshall/v3/ActiveMQT
3470 3354

src/main/activemq/wireformat/openwire/marshalsrv2/ProducerIdMarshaller/openwire/marshall/v3/ActiveMQT
3473 3358

src/main/activemq/wireformat/openwire/marshalsrv2/ProducerInfoMarshaller/openwire/marshall/v3/BrokerIdMa
3476 3361

src/main/activemq/wireformat/openwire/marshalsrv2/ProducerInfoMarshaller/openwire/marshall/v3/BrokerInfoM
3480 3364

src/main/activemq/wireformat/openwire/marshalsrv2/ProducerSubscriptionMarshaller/openwire/marshall/v3/Connection
3483 3368

src/main/activemq/wireformat/openwire/marshalsrv2/ReplyCommandMarshaller/openwire/marshall/v3/Connection
3486 3371

src/main/activemq/wireformat/openwire/marshalsrv2/ResponsiveMarshaller/openwire/marshall/v3/Connection
3490 3374

src/main/activemq/wireformat/openwire/marshalsrv2/SessionIdMarshaller/openwire/marshall/v3/Connection
3493 3378

src/main/activemq/wireformat/openwire/marshalsrv2/SessionInfoMarshaller/openwire/marshall/v3/ConsumerC
3496 3381

src/main/activemq/wireformat/openwire/marshalsrv2/SinkAndSinkInfoMarshaller/openwire/marshall/v3/ConsumerId
3500 3384

src/main/activemq/wireformat/openwire/marshalsrv2/SinkAndSinkInfoMarshaller/openwire/marshall/v3/ConsumerIn
3503 3388

src/main/activemq/wireformat/openwire/marshalsrv2/SinkAndSinkInfoMarshaller/openwire/marshall/v3/ControlCom
3506 3391

src/main/activemq/wireformat/openwire/marshalsrv2/SinkAndSinkInfoMarshaller/openwire/marshall/v3/DataArrayF
3510 3394

src/main/activemq/wireformat/openwire/marshalsrv2/WireFormatInfoMarshaller/openwire/marshall/v3/DataRespon
3513 3398

src/main/activemq/wireformat/openwire/marshalsrv2/KafkaSinkAndSinkInfoMarshaller/openwire/marshall/v3/Destination
3516 3401

src/main/activemq/wireformat/openwire/marshalsrv3/main/activemq/BlockMessageMarshaller/openwire/marshall/v3/DiscoveryE
3314 3404

src/main/activemq/wireformat/openwire/marshalsrv3/main/activemq/BytesMessageMarshaller/openwire/marshall/v3/ExceptionR
3318 3408

src/main/activemq/wireformat/openwire/marshalsrv3/main/activemq/QueueWireFormatMarshaller/openwire/marshall/v3/FlushComm
3321 3411

src/main/activemq/wireformat/openwire/marshalsrv3/main/activemq/QueueWireFormatMarshaller/openwire/marshall/v3/IntegerResp
3324 3414

src/main/activemq/wireformat/openwire/marshalsrv3/main/activemq/QueueWireFormatMarshaller/openwire/marshall/v3/JournalQue
3328 3418

src/main/activemq/wireformat/openwire/marshalsrv3/main/activemq/QueueWireFormatMarshaller/openwire/marshall/v3/JournalTop
3331 3421

src/main/activemq/wireformat/openwire/marshalsrv3/main/activemq/QueueWireFormatMarshaller/openwire/marshall/v3/JournalTrac
3334 3424

src/main/activemq/wireformat/openwire/marshalsrv3/main/activemq/QueueWireFormatMarshaller/openwire/marshall/v3/JournalTran
3338 3428

src/main/activemq/wireformat/openwire/marshalsrv3/main/activemq/QueueWireFormatMarshaller/openwire/marshall/v3/KeepAliveIn
3341 3431

src/main/activemq/wireformat/openwire/marshaller/v3/LastPartialCommandMarshaller/openwire/marshaller/v4/ActiveMQB	3434	3318
src/main/activemq/wireformat/openwire/marshaller/v3/LocalTransactionIdMarshaller/openwire/marshaller/v4/ActiveMQD	3438	3322
src/main/activemq/wireformat/openwire/marshaller/v3/MessageHeaderFactory/openwire/marshaller/v4/ActiveMQM	3441	3325
src/main/activemq/wireformat/openwire/marshaller/v3/MessageHeaderMarshaller/openwire/marshaller/v4/ActiveMQM	3443	3328
src/main/activemq/wireformat/openwire/marshaller/v3/MessageDispatchInfoMarshaller/openwire/marshaller/v4/ActiveMQC	3447	3332
src/main/activemq/wireformat/openwire/marshaller/v3/MessageDispatchNotificationMarshaller/openwire/marshaller/v4/ActiveMQC	3450	3335
src/main/activemq/wireformat/openwire/marshaller/v3/MessageHeaderMarshaller/openwire/marshaller/v4/ActiveMQS	3454	3338
src/main/activemq/wireformat/openwire/marshaller/v3/MessageHeaderMarshaller/openwire/marshaller/v4/ActiveMQT	3457	3342
src/main/activemq/wireformat/openwire/marshaller/v3/MessageHeaderMarshaller/openwire/marshaller/v4/ActiveMQT	3460	3345
src/main/activemq/wireformat/openwire/marshaller/v3/NetworkBridgeFilterMarshaller/openwire/marshaller/v4/ActiveMQT	3464	3348
src/main/activemq/wireformat/openwire/marshaller/v3/PartialCommandMarshaller/openwire/marshaller/v4/ActiveMQT	3467	3352
src/main/activemq/wireformat/openwire/marshaller/v3/ProducerIdMarshaller/openwire/marshaller/v4/ActiveMQT	3470	3355
src/main/activemq/wireformat/openwire/marshaller/v3/ProducerIdMarshaller/openwire/marshaller/v4/BaseComm	3474	3358
src/main/activemq/wireformat/openwire/marshaller/v3/ProducerIdMarshaller/openwire/marshaller/v4/BrokerIdMa	3477	3362
src/main/activemq/wireformat/openwire/marshaller/v3/ProducerInfoMarshaller/openwire/marshaller/v4/BrokerInfo	3480	3365
src/main/activemq/wireformat/openwire/marshaller/v3/ProducerSubscriptionInfoMarshaller/openwire/marshaller/v4/Connection	3484	3368
src/main/activemq/wireformat/openwire/marshaller/v3/ReplyCommandMarshaller/openwire/marshaller/v4/Connection	3487	3372
src/main/activemq/wireformat/openwire/marshaller/v3/ReplyHeaderMarshaller/openwire/marshaller/v4/Connection	3490	3375
src/main/activemq/wireformat/openwire/marshaller/v3/SessionIdMarshaller/openwire/marshaller/v4/Connection	3494	3378
src/main/activemq/wireformat/openwire/marshaller/v3/SessionInfoMarshaller/openwire/marshaller/v4/ConsumerC	3497	3382
src/main/activemq/wireformat/openwire/marshaller/v3/ShutdownInfoMarshaller/openwire/marshaller/v4/ConsumerId	3500	3385
src/main/activemq/wireformat/openwire/marshaller/v3/SubscriptionInfoMarshaller/openwire/marshaller/v4/ConsumerIn	3504	3388
src/main/activemq/wireformat/openwire/marshaller/v3/TransactionIdMarshaller/openwire/marshaller/v4/ControlCom	3507	3392
src/main/activemq/wireformat/openwire/marshaller/v3/TransactionInfoMarshaller/openwire/marshaller/v4/DataArrayF	3510	3395
src/main/activemq/wireformat/openwire/marshaller/v3/WireFormatInfoMarshaller/openwire/marshaller/v4/DataRespon	3514	3398
src/main/activemq/wireformat/openwire/marshaller/v3/XML/activemq/IdMarshaller/openwire/marshaller/v4/Destination	3517	3402
src/main/activemq/wireformat/openwire/marshaller/v4/ActiveMQBlockMessageMarshaller/openwire/marshaller/v4/DiscoveryE	3315	3405

src/main/activemq/wireformat/openwire/marshaller/v4/ExceptionResponseMarshaller/openwire/marshaller/v4/SessionInfo	3408	3498
src/main/activemq/wireformat/openwire/marshaller/v4/FlushActiveMQWireFormat/openwire/marshaller/v4/ShutDownIn	3412	3501
src/main/activemq/wireformat/openwire/marshaller/v4/IntegerResponseMarshaller/openwire/marshaller/v4/Subscription	3415	3504
src/main/activemq/wireformat/openwire/marshaller/v4/JainActiveMQWireFormat/openwire/marshaller/v4/Transaction	3418	3508
src/main/activemq/wireformat/openwire/marshaller/v4/JainActiveMQWireFormat/openwire/marshaller/v4/Transaction	3422	3511
src/main/activemq/wireformat/openwire/marshaller/v4/JainActiveMQWireFormat/openwire/marshaller/v4/WireFormat	3425	3514
src/main/activemq/wireformat/openwire/marshaller/v4/JainActiveMQWireFormat/openwire/marshaller/v4/XATransact	3428	3518
src/main/activemq/wireformat/openwire/marshaller/v4/KeepActiveInfoMarshaller/openwire/marshaller/v5/ActiveMQB	3432	3316
src/main/activemq/wireformat/openwire/marshaller/v4/LastPartialConnectionMarshaller/openwire/marshaller/v5/ActiveMQB	3435	3319
src/main/activemq/wireformat/openwire/marshaller/v4/LocalTransactionInfoMarshaller/openwire/marshaller/v5/ActiveMQD	3438	3322
src/main/activemq/wireformat/openwire/marshaller/v4/MarshallerFactory/openwire/marshaller/v5/ActiveMQM	3441	3326
src/main/activemq/wireformat/openwire/marshaller/v4/MisplacedMarshaller/openwire/marshaller/v5/ActiveMQM	3444	3329
src/main/activemq/wireformat/openwire/marshaller/v4/MisplacedDispatcherMarshaller/openwire/marshaller/v5/ActiveMQC	3447	3332
src/main/activemq/wireformat/openwire/marshaller/v4/MisplacedDispatcherMarshaller/openwire/marshaller/v5/ActiveMQC	3451	3336
src/main/activemq/wireformat/openwire/marshaller/v4/MisplacedMarshaller/openwire/marshaller/v5/ActiveMQS	3454	3339
src/main/activemq/wireformat/openwire/marshaller/v4/MisplacedMarshaller/openwire/marshaller/v5/ActiveMQT	3458	3342
src/main/activemq/wireformat/openwire/marshaller/v4/MisplacedMarshaller/openwire/marshaller/v5/ActiveMQT	3461	3346
src/main/activemq/wireformat/openwire/marshaller/v4/NewBridgeFilterMarshaller/openwire/marshaller/v5/ActiveMQT	3464	3349
src/main/activemq/wireformat/openwire/marshaller/v4/PartialConnectionMarshaller/openwire/marshaller/v5/ActiveMQT	3468	3352
src/main/activemq/wireformat/openwire/marshaller/v4/ProducerAckMarshaller/openwire/marshaller/v5/ActiveMQT	3471	3356
src/main/activemq/wireformat/openwire/marshaller/v4/ProducerIdMarshaller/openwire/marshaller/v5/BaseComm	3474	3359
src/main/activemq/wireformat/openwire/marshaller/v4/ProducerIdMarshaller/openwire/marshaller/v5/BrokerIdMa	3478	3362
src/main/activemq/wireformat/openwire/marshaller/v4/ProducerInfoMarshaller/openwire/marshaller/v5/BrokerInfoM	3481	3366
src/main/activemq/wireformat/openwire/marshaller/v4/ProducerSubscriptionInfoMarshaller/openwire/marshaller/v5/Connection	3484	3369
src/main/activemq/wireformat/openwire/marshaller/v4/ReplyConnectionMarshaller/openwire/marshaller/v5/Connection	3488	3372
src/main/activemq/wireformat/openwire/marshaller/v4/ResponsiveMarshaller/openwire/marshaller/v5/Connection	3491	3376
src/main/activemq/wireformat/openwire/marshaller/v4/SessionIdMarshaller/openwire/marshaller/v5/Connection	3494	3379

src/main/activemq/wireformat/openwire/marshaller/v5/ConsumerGroupWireMarshal/openwire/marshaller/v5/ProducerAck	3382	3472
src/main/activemq/wireformat/openwire/marshaller/v5/ConsumerIdWireMarshal/openwire/marshaller/v5/ProducerId	3386	3475
src/main/activemq/wireformat/openwire/marshaller/v5/ConsumerInfoWireMarshal/openwire/marshaller/v5/ProducerInfo	3389	3478
src/main/activemq/wireformat/openwire/marshaller/v5/ConsumerQueueWireMarshal/openwire/marshaller/v5/RemoveInfo	3392	3482
src/main/activemq/wireformat/openwire/marshaller/v5/DataActiveResponseWireMarshal/openwire/marshaller/v5/RemoveSub	3396	3485
src/main/activemq/wireformat/openwire/marshaller/v5/DataResponseWireMarshal/openwire/marshaller/v5/ReplayCom	3399	3488
src/main/activemq/wireformat/openwire/marshaller/v5/DestinationInfoWireMarshal/openwire/marshaller/v5/ResponseM	3402	3492
src/main/activemq/wireformat/openwire/marshaller/v5/DisconnectWireMarshal/openwire/marshaller/v5/SessionIdM	3406	3495
src/main/activemq/wireformat/openwire/marshaller/v5/ExceptionResponseWireMarshal/openwire/marshaller/v5/SessionInfo	3409	3498
src/main/activemq/wireformat/openwire/marshaller/v5/FlushActiveQueueWireMarshal/openwire/marshaller/v5/ShutDownIn	3412	3502
src/main/activemq/wireformat/openwire/marshaller/v5/IntegerResponseWireMarshal/openwire/marshaller/v5/Subscription	3416	3505
src/main/activemq/wireformat/openwire/marshaller/v5/MainActiveQueueWireMarshal/openwire/marshaller/v5/Transaction	3419	3508
src/main/activemq/wireformat/openwire/marshaller/v5/MainActiveQueueWireMarshal/openwire/marshaller/v5/Transaction	3422	3512
src/main/activemq/wireformat/openwire/marshaller/v5/MainActiveQueueWireMarshal/openwire/marshaller/v5/WireFormat	3426	3515
src/main/activemq/wireformat/openwire/marshaller/v5/MainActiveQueueWireMarshal/openwire/marshaller/v5/XATransact	3429	3518
src/main/activemq/wireformat/openwire/marshaller/v5/MainActiveQueueWireMarshal/openwire/OpenWireFormat.h,	3432	3519
src/main/activemq/wireformat/openwire/marshaller/v5/MainActiveQueueWireMarshal/openwire/OpenWireFormatFactory	3436	3520
src/main/activemq/wireformat/openwire/marshaller/v5/MainActiveQueueWireMarshal/openwire/OpenWireFormatNegoti	3439	3520
src/main/activemq/wireformat/openwire/marshaller/v5/MainActiveQueueWireMarshal/openwire/OpenWireResponseBuild	3442	3521
src/main/activemq/wireformat/openwire/marshaller/v5/MainActiveQueueWireMarshal/openwire/Utils/BooleanStream.h,	3445	3521
src/main/activemq/wireformat/openwire/marshaller/v5/MainActiveQueueWireMarshal/openwire/Utils/HexTable.h,	3448	3522
src/main/activemq/wireformat/openwire/marshaller/v5/MainActiveQueueWireMarshal/openwire/Utils/MessagePropertyIn	3452	3523
src/main/activemq/wireformat/openwire/marshaller/v5/MainActiveQueueWireMarshal/openwire/Utils/OpenwireStringSup	3455	3523
src/main/activemq/wireformat/openwire/marshaller/v5/MainActiveQueueWireMarshal/openwire/Utils/StompCommandConstants	3458	3524
src/main/activemq/wireformat/openwire/marshaller/v5/MainActiveQueueWireMarshal/openwire/Utils/StompFrame.h,	3462	3524
src/main/activemq/wireformat/openwire/marshaller/v5/MainActiveQueueWireMarshal/openwire/Utils/StompHelper.h,	3465	3525
src/main/activemq/wireformat/openwire/marshaller/v5/MainActiveQueueWireMarshal/openwire/Utils/StompWireFormat.h,	3468	3525

- src/main/activemq/wireformat/stomp/StompWireFormat.h, 3547
- src/main/activemq/wireformat/WireFormat.h, 3527
- src/main/activemq/wireformat/WireFormatFactory.h, 3527
- src/main/activemq/wireformat/WireFormatNegotiator.h, 3528
- src/main/activemq/wireformat/WireFormatRegistry.h, 3528
- src/main/cms/BytesMessage.h, 3529
- src/main/cms/Closeable.h, 3529
- src/main/cms/CMSException.h, 3530
- src/main/cms/CMSProperties.h, 3531
- src/main/cms/CMSSecurityException.h, 3531
- src/main/cms/Config.h, 3306
- src/main/cms/Connection.h, 3532
- src/main/cms/ConnectionFactory.h, 3532
- src/main/cms/ConnectionMetaData.h, 3533
- src/main/cms/DeliveryMode.h, 3533
- src/main/cms/Destination.h, 3533
- src/main/cms/ExceptionListener.h, 3534
- src/main/cms/IllegalStateException.h, 3534
- src/main/cms/InvalidClientIdException.h, 3535
- src/main/cms/InvalidDestinationException.h, 3536
- src/main/cms/InvalidSelectorException.h, 3536
- src/main/cms/MapMessage.h, 3536
- src/main/cms/Message.h, 3249
- src/main/cms/MessageConsumer.h, 3537
- src/main/cms/MessageEOFException.h, 3537
- src/main/cms/MessageFormatException.h, 3538
- src/main/cms/MessageListener.h, 3538
- src/main/cms/MessageNotReadableException.h, 3539
- src/main/cms/MessageNotWritableException.h, 3539
- src/main/cms/MessageProducer.h, 3540
- src/main/cms/ObjectMessage.h, 3540
- src/main/cms/Queue.h, 3541
- src/main/cms/QueueBrowser.h, 3542
- src/main/cms/Session.h, 3542
- src/main/cms/Startable.h, 3543
- src/main/cms/Stopable.h, 3543
- src/main/cms/StreamMessage.h, 3544
- src/main/cms/TemporaryQueue.h, 3544
- src/main/cms/TemporaryTopic.h, 3545
- src/main/cms/TextMessage.h, 3545
- src/main/cms/Topic.h, 3546
- src/main/cms/UnsupportedOperationException.h, 3546
- src/main/decaf/internal/AprPool.h, 3547
- src/main/decaf/internal/DecafRuntime.h, 3547
- src/main/decaf/internal/io/StandardErrorOutputStream.h, 3548
- src/main/decaf/internal/io/StandardInputStream.h, 3548
- src/main/decaf/internal/io/StandardOutputStream.h, 3548
- src/main/decaf/internal/net/URLEncoderDecoder.h, 3549
- src/main/decaf/internal/net/URIHelper.h, 3550
- src/main/decaf/internal/net/URIType.h, 3550
- src/main/decaf/internal/nio/BufferFactory.h, 3551
- src/main/decaf/internal/nio/ByteBuffer.h, 3551
- src/main/decaf/internal/nio/ByteBufferPerspective.h, 3552
- src/main/decaf/internal/nio/CharArrayBuffer.h, 3552
- src/main/decaf/internal/nio/DoubleArrayBuffer.h, 3553
- src/main/decaf/internal/nio/FloatArrayBuffer.h, 3554
- src/main/decaf/internal/nio/IntArrayBuffer.h, 3554
- src/main/decaf/internal/nio/LongArrayBuffer.h, 3555
- src/main/decaf/internal/nio/ShortArrayBuffer.h, 3555
- src/main/decaf/internal/util/ByteArrayAdapter.h, 3556
- src/main/decaf/internal/util/concurrent/ConditionImpl.h, 3556
- src/main/decaf/internal/util/concurrent/MutexImpl.h, 3557
- src/main/decaf/internal/util/concurrent/SynchronizableImpl.h, 3557
- src/main/decaf/internal/util/concurrent/Transferer.h, 3558
- src/main/decaf/internal/util/concurrent/TransferQueue.h, 3558
- src/main/decaf/internal/util/concurrent/TransferStack.h, 3559
- src/main/decaf/internal/util/concurrent/unix/ConditionHandle.h, 3559
- src/main/decaf/internal/util/concurrent/unix/MutexHandle.h, 3560
- src/main/decaf/internal/util/concurrent/windows/ConditionHandle.h, 3560
- src/main/decaf/internal/util/concurrent/windows/MutexHandle.h, 3561
- src/main/decaf/internal/util/HexStringParser.h, 3561

- src/main/decaf/internal/util/TimerTaskHeap.h, 3561
- src/main/decaf/io/BlockingByteArrayInputStream.h, 3562
- src/main/decaf/io/BufferedInputStream.h, 3562
- src/main/decaf/io/BufferedOutputStream.h, 3563
- src/main/decaf/io/ByteArrayInputStream.h, 3563
- src/main/decaf/io/ByteArrayOutputStream.h, 3564
- src/main/decaf/io/Closeable.h, 3530
- src/main/decaf/io/DataInputStream.h, 3564
- src/main/decaf/io/DataOutputStream.h, 3565
- src/main/decaf/io/EOFException.h, 3565
- src/main/decaf/io/FilterInputStream.h, 3566
- src/main/decaf/io/FilterOutputStream.h, 3566
- src/main/decaf/io/InputStream.h, 3567
- src/main/decaf/io/InterruptedIOException.h, 3567
- src/main/decaf/io/IOException.h, 3568
- src/main/decaf/io/OutputStream.h, 3568
- src/main/decaf/io/Reader.h, 3569
- src/main/decaf/io/UnsupportedEncodingException.h, 3569
- src/main/decaf/io/UTFDataFormatException.h, 3569
- src/main/decaf/io/Writer.h, 3570
- src/main/decaf/lang/Appendable.h, 3570
- src/main/decaf/lang/Boolean.h, 3571
- src/main/decaf/lang/Byte.h, 3571
- src/main/decaf/lang/Character.h, 3572
- src/main/decaf/lang/CharSequence.h, 3572
- src/main/decaf/lang/Comparable.h, 3572
- src/main/decaf/lang/Double.h, 3573
- src/main/decaf/lang/Exception.h, 3573
- src/main/decaf/lang/exceptions/ClassCastException.h, 3574
- src/main/decaf/lang/exceptions/ExceptionDefinition.h, 3274
- src/main/decaf/lang/exceptions/IllegalArgumentException.h, 3574
- src/main/decaf/lang/exceptions/IllegalMonitorStateException.h, 3575
- src/main/decaf/lang/exceptions/IllegalStateException.h, 3535
- src/main/decaf/lang/exceptions/IllegalThreadStateException.h, 3575
- src/main/decaf/lang/exceptions/IndexOutOfBoundsException.h, 3575
- src/main/decaf/lang/exceptions/InterruptedException.h, 3576
- src/main/decaf/lang/exceptions/InvalidStateException.h, 3576
- src/main/decaf/lang/exceptions/NoSuchElementException.h, 3577
- src/main/decaf/lang/exceptions/NullPointerException.h, 3577
- src/main/decaf/lang/exceptions/NumberFormatException.h, 3577
- src/main/decaf/lang/exceptions/RuntimeException.h, 3578
- src/main/decaf/lang/exceptions/UnsupportedOperationException.h, 3546
- src/main/decaf/lang/Float.h, 3578
- src/main/decaf/lang/Integer.h, 3579
- src/main/decaf/lang/Iterable.h, 3579
- src/main/decaf/lang/Long.h, 3579
- src/main/decaf/lang/Math.h, 3580
- src/main/decaf/lang/Number.h, 3580
- src/main/decaf/lang/Pointer.h, 3581
- src/main/decaf/lang/Runnable.h, 3582
- src/main/decaf/lang/Runtime.h, 3582
- src/main/decaf/lang/Short.h, 3583
- src/main/decaf/lang/System.h, 3583
- src/main/decaf/lang/Thread.h, 3583
- src/main/decaf/lang/ThreadGroup.h, 3584
- src/main/decaf/lang/Throwable.h, 3584
- src/main/decaf/net/BindException.h, 3585
- src/main/decaf/net/BufferedSocket.h, 3585
- src/main/decaf/net/ConnectException.h, 3586
- src/main/decaf/net/HttpRetryException.h, 3586
- src/main/decaf/net/MalformedURLException.h, 3587
- src/main/decaf/net/NoRouteToHostException.h, 3587
- src/main/decaf/net/PortUnreachableException.h, 3587
- src/main/decaf/net/ProtocolException.h, 3588
- src/main/decaf/net/ServerSocket.h, 3588
- src/main/decaf/net/Socket.h, 3589
- src/main/decaf/net/SocketError.h, 3589
- src/main/decaf/net/SocketException.h, 3590
- src/main/decaf/net/SocketFactory.h, 3590
- src/main/decaf/net/SocketInputStream.h, 3591
- src/main/decaf/net/SocketOutputStream.h, 3591
- src/main/decaf/net/SocketTimeoutException.h, 3591
- src/main/decaf/net/TcpSocket.h, 3592
- src/main/decaf/net/UnknownHostException.h, 3593
- src/main/decaf/net/UnknownServiceException.h, 3593

- src/main/decaf/net/URI.h, 3593
- src/main/decaf/net/URISyntaxException.h, 3594
- src/main/decaf/net/URL.h, 3594
- src/main/decaf/net/URLDecoder.h, 3595
- src/main/decaf/net/URLEncoder.h, 3595
- src/main/decaf/nio/Buffer.h, 3596
- src/main/decaf/nio/BufferOverflowException.h, 3596
- src/main/decaf/nio/BufferUnderflowException.h, 3596
- src/main/decaf/nio/ByteBuffer.h, 3597
- src/main/decaf/nio/CharBuffer.h, 3597
- src/main/decaf/nio/DoubleBuffer.h, 3598
- src/main/decaf/nio/FloatBuffer.h, 3599
- src/main/decaf/nio/IntBuffer.h, 3599
- src/main/decaf/nio/InvalidMarkException.h, 3600
- src/main/decaf/nio/LongBuffer.h, 3600
- src/main/decaf/nio/ReadOnlyBufferException.h, 3601
- src/main/decaf/nio/ShortBuffer.h, 3601
- src/main/decaf/security/auth/x500/X500Principal.h, 3602
- src/main/decaf/security/cert/Certificate.h, 3602
- src/main/decaf/security/cert/CertificateEncodingException.h, 3603
- src/main/decaf/security/cert/CertificateException.h, 3603
- src/main/decaf/security/cert/CertificateExpiredException.h, 3604
- src/main/decaf/security/cert/CertificateNotYetValidException.h, 3604
- src/main/decaf/security/cert/CertificateParsingException.h, 3604
- src/main/decaf/security/cert/X509Certificate.h, 3605
- src/main/decaf/security/GeneralSecurityException.h, 3605
- src/main/decaf/security/InvalidKeyException.h, 3606
- src/main/decaf/security/Key.h, 3606
- src/main/decaf/security/KeyException.h, 3607
- src/main/decaf/security/NoSuchAlgorithmException.h, 3607
- src/main/decaf/security/NoSuchProviderException.h, 3607
- src/main/decaf/security/Principal.h, 3608
- src/main/decaf/security/PublicKey.h, 3608
- src/main/decaf/security/SignatureException.h, 3609
- src/main/decaf/security_ - provider/SecurityProvider.h, 3609
- src/main/decaf/security_ - provider/SecurityProviderMap.h, 3609
- src/main/decaf/security_ - provider/SecurityProviderRegistrar.h, 3610
- src/main/decaf/security_ - provider/unix/openssl/OpenSSLX500Principal.h, 3610
- src/main/decaf/security_ - provider/unix/openssl/OpenSSLX509Certificate.h, 3611
- src/main/decaf/util/AbstractCollection.h, 3611
- src/main/decaf/util/AbstractList.h, 3612
- src/main/decaf/util/AbstractMap.h, 3612
- src/main/decaf/util/AbstractQueue.h, 3613
- src/main/decaf/util/AbstractSequentialList.h, 3614
- src/main/decaf/util/AbstractSet.h, 3614
- src/main/decaf/util/Collection.h, 3615
- src/main/decaf/util/Comparator.h, 3615
- src/main/decaf/util/comparators/Less.h, 3616
- src/main/decaf/util/concurrent/atomic/AtomicBoolean.h, 3616
- src/main/decaf/util/concurrent/atomic/AtomicInteger.h, 3617
- src/main/decaf/util/concurrent/atomic/AtomicReference.h, 3617
- src/main/decaf/util/concurrent/BlockingQueue.h, 3618
- src/main/decaf/util/concurrent/BrokenBarrierException.h, 3618
- src/main/decaf/util/concurrent/Callable.h, 3619
- src/main/decaf/util/concurrent/CancellationException.h, 3619
- src/main/decaf/util/concurrent/Concurrent.h, 3619
- src/main/decaf/util/concurrent/ConcurrentMap.h, 3620
- src/main/decaf/util/concurrent/ConcurrentStlMap.h, 3621
- src/main/decaf/util/concurrent/CountDownLatch.h, 3622
- src/main/decaf/util/concurrent/Delayed.h, 3622
- src/main/decaf/util/concurrent/ExecutionException.h, 3623
- src/main/decaf/util/concurrent/Executor.h, 3623
- src/main/decaf/util/concurrent/ExecutorService.h, 3623
- src/main/decaf/util/concurrent/Future.h, 3624
- src/main/decaf/util/concurrent/Lock.h, 3624

- src/main/decaf/util/concurrent/locks/Condition.h, 3625
- src/main/decaf/util/concurrent/locks/Lock.h, 3625
- src/main/decaf/util/concurrent/locks/LockSupport.h, 3626
- src/main/decaf/util/concurrent/locks/ReadWriteLock.h, 3627
- src/main/decaf/util/concurrent/locks/ReentrantLock.h, 3627
- src/main/decaf/util/concurrent/Mutex.h, 3628
- src/main/decaf/util/concurrent/PooledThread.h, 3628
- src/main/decaf/util/concurrent/PooledThreadListener.h, 3629
- src/main/decaf/util/concurrent/RejectedExecutionException.h, 3629
- src/main/decaf/util/concurrent/RejectedExecutionHandler.h, 3629
- src/main/decaf/util/concurrent/Semaphore.h, 3630
- src/main/decaf/util/concurrent/Synchronizable.h, 3630
- src/main/decaf/util/concurrent/SynchronousQueue.h, 3631
- src/main/decaf/util/concurrent/TaskListener.h, 3632
- src/main/decaf/util/concurrent/ThreadFactory.h, 3632
- src/main/decaf/util/concurrent/ThreadPool.h, 3632
- src/main/decaf/util/concurrent/TimeoutException.h, 3633
- src/main/decaf/util/concurrent/TimeUnit.h, 3634
- src/main/decaf/util/Config.h, 3306
- src/main/decaf/util/Date.h, 3634
- src/main/decaf/util/Iterator.h, 3635
- src/main/decaf/util/List.h, 3635
- src/main/decaf/util/ListIterator.h, 3636
- src/main/decaf/util/logging/ConsoleHandler.h, 3636
- src/main/decaf/util/logging/Filter.h, 3636
- src/main/decaf/util/logging/Formatter.h, 3637
- src/main/decaf/util/logging/Handler.h, 3637
- src/main/decaf/util/logging/Logger.h, 3638
- src/main/decaf/util/logging/LoggerCommon.h, 3638
- src/main/decaf/util/logging/LoggerDefines.h, 3639
- src/main/decaf/util/logging/LoggerHierarchy.h, 3640
- src/main/decaf/util/logging/LogManager.h, 3640
- src/main/decaf/util/logging/LogRecord.h, 3641
- src/main/decaf/util/logging/LogWriter.h, 3641
- src/main/decaf/util/logging/MarkBlockLogger.h, 3642
- src/main/decaf/util/logging/PropertiesChangeListener.h, 3642
- src/main/decaf/util/logging/SimpleFormatter.h, 3643
- src/main/decaf/util/logging/SimpleLogger.h, 3643
- src/main/decaf/util/logging/StreamHandler.h, 3644
- src/main/decaf/util/Map.h, 3644
- src/main/decaf/util/PriorityQueue.h, 3645
- src/main/decaf/util/Properties.h, 3645
- src/main/decaf/util/Queue.h, 3541
- src/main/decaf/util/Random.h, 3646
- src/main/decaf/util/Set.h, 3647
- src/main/decaf/util/StlList.h, 3647
- src/main/decaf/util/StlMap.h, 3648
- src/main/decaf/util/StlQueue.h, 3648
- src/main/decaf/util/StlSet.h, 3649
- src/main/decaf/util/StringTokenizer.h, 3650
- src/main/decaf/util/Timer.h, 3650
- src/main/decaf/util/TimerTask.h, 3651
- src/main/decaf/util/UUID.h, 3651
- stackTrace
 - decaf::lang::Exception, 1482
- StandardErrorOutputStream
 - decaf::internal::io::StandardErrorOutputStream, 2814
- StandardInputStream
 - decaf::internal::io::StandardInputStream, 2820
- StandardOutputStream
 - decaf::internal::io::StandardOutputStream, 2826
- start
 - activemq::commands::ConsumerControl, 1170
 - activemq::core::ActiveMQConnection, 210
 - activemq::core::ActiveMQConsumer, 237
 - activemq::core::ActiveMQSession, 417
 - activemq::core::ActiveMQSessionExecutor, 421
 - activemq::core::MessageDispatchChannel, 2093
 - activemq::transport::correlator::ResponseCorrelator, 2619
 - activemq::transport::failover::FailoverTransport, 1522
 - activemq::transport::IOTransport, 1715

- activemq::transport::mock::MockTransport, 2236
- activemq::transport::Transport, 3070
- activemq::transport::TransportFilter, 3079
- activemq::wireformat::openwire::OpenWireFormat, 2313
- cms::Startable, 2831
- startupTransport
 - activemq::core::ActiveMQConnectionSupport, 226
- staticCast
 - decaf::lang::Pointer, 2349
- StaticInitializer
 - activemq::core::ActiveMQConstants::StaticInitializer, 2832
- std, 117
- std::binary_function, 663
- std::less< decaf::lang::Pointer< T > >, 1864
- operator(), 1865
- StlList
 - decaf::util::StlList, 2837, 2838
- StlMap
 - decaf::util::StlMap, 2849
- StlQueue
 - decaf::util::StlQueue, 2860
- StlSet
 - decaf::util::StlSet, 2868
- StompFrame
 - activemq::wireformat::stomp::StompFrame, 2875
- StompHelper
 - activemq::wireformat::stomp::StompHelper, 2880
- StompWireFormat
 - activemq::wireformat::stomp::StompWireFormat, 2884
- StompWireFormatFactory
 - activemq::wireformat::stomp::StompWireFormatFactory, 2887
- stop
 - activemq::commands::ConsumerControl, 1170
 - activemq::core::ActiveMQConnection, 210
 - activemq::core::ActiveMQConsumer, 238
 - activemq::core::ActiveMQSession, 417
 - activemq::core::ActiveMQSessionExecutor, 421
 - activemq::core::MessageDispatchChannel, 2093
 - activemq::transport::failover::FailoverTransport, 1522
 - activemq::transport::IOTransport, 1715
 - activemq::transport::mock::MockTransport, 2236
 - activemq::transport::Transport, 3071
 - activemq::transport::TransportFilter, 3080
 - cms::Stoppable, 2888
 - decaf::util::concurrent::PooledThread, 2365
 - decaf::util::Negotiator, 2501, 2502
 - decaf::util::Properties, 2501, 2502
 - StreamHandler
 - decaf::util::logging::StreamHandler, 2889
 - STRING_TYPE
 - activemq::util::PrimitiveValueNode, 2402
 - StringTokenizer
 - decaf::util::StringTokenizer, 2905
 - stringValue
 - activemq::util::PrimitiveValueNode::PrimitiveValue, 2396
 - subscriptionName
 - activemq::commands::RemoveSubscriptionInfo, 2564
 - activemq::commands::SubscriptionInfo, 2911
 - SUBSCRIBE
 - activemq::wireformat::stomp::StompCommandConstants, 2873
 - subscribedDestination
 - activemq::commands::SubscriptionInfo, 2911
 - SubscriptionInfo
 - activemq::commands::SubscriptionInfo, 2908
 - SubscriptionInfoMarshaller
 - activemq::wireformat::openwire::marshal::v1::SubscriptionInfo, 2912
 - activemq::wireformat::openwire::marshal::v2::SubscriptionInfo, 2928
 - activemq::wireformat::openwire::marshal::v3::SubscriptionInfo, 2916
 - activemq::wireformat::openwire::marshal::v4::SubscriptionInfo, 2924
 - activemq::wireformat::openwire::marshal::v5::SubscriptionInfo, 2920
 - subscriptionName
 - activemq::commands::ConsumerInfo, 1222
 - subscriptionName
 - activemq::commands::JournalTopicAck, 1746
 - subSequence
 - decaf::internal::nio::CharArrayBuffer, 929
 - decaf::lang::CharSequence, 947
 - decaf::nio::CharBuffer, 945
 - suspend
 - activemq::commands::ConnectionControl, 1059
 - swap
 - decaf::lang::AtomicRefCounter, 589

- decaf::lang::Pointer, 2349
- SynchronizableImpl
 - decaf::internal::util::concurrent::SynchronizableImpl, 2873
 - 2943
- synchronized
 - Concurrent.h, 3620
- SynchronousQueue
 - decaf::util::concurrent::SynchronousQueue, 2949
- syncRequest
 - activemq::core::ActiveMQConnection, 211
 - activemq::core::ActiveMQSession, 417
- System
 - decaf::lang::System, 2954
- take
 - decaf::util::concurrent::SynchronousQueue, 2952
- takeRef
 - decaf::internal::nio::ByteArrayPerspective, 847
- takeSession
 - activemq::cmsutil::SessionPool, 2726
- targetConsumerId
 - activemq::commands::Message, 2035
- Task
 - decaf::util::concurrent::ThreadPool, 2979
- TcpSocket
 - decaf::net::TcpSocket, 2961
- TcpTransport
 - activemq::transport::tcp::TcpTransport, 2968
- TEMP_POSTFIX
 - activemq::commands::ActiveMQDestination, 250
- TEMP_PREFIX
 - activemq::commands::ActiveMQDestination, 250
- TEMP_QUEUE_QUALIFIED_PREFIX
 - activemq::commands::ActiveMQDestination, 250
- TEMP_TOPIC_QUALIFIED_PREFIX
 - activemq::commands::ActiveMQDestination, 250
- TEMPORARY_QUEUE
 - cms::Destination, 1388
- TEMPORARY_TOPIC
 - cms::Destination, 1388
- TEMPQUEUE_PREFIX
 - activemq::wireformat::stomp::StompCommandConstants, 2873
- TEMPTOPIC_PREFIX
 - activemq::wireformat::stomp::StompCommandConstants, 2873
- TEXT
 - activemq::wireformat::stomp::StompCommandConstants, 529
- text
 - activemq::commands::ActiveMQTextMessage, 529
- ThreadGroup
 - decaf::lang::ThreadGroup, 2977
- ThreadPool
 - decaf::util::concurrent::ThreadPool, 2979
- Throwable
 - decaf::lang::Throwable, 2984
- Throwing
 - decaf::util::logging, 117
- tightMarshal
 - activemq::wireformat::openwire::marshal::BaseDataStreamMarshal, 648
 - activemq::wireformat::openwire::marshal::DataStreamMarshal, 1354
 - activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMarshal, 153
 - activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMarshal, 188
 - activemq::wireformat::openwire::marshal::v1::ActiveMQDestination, 256
 - activemq::wireformat::openwire::marshal::v1::ActiveMQMapMarshal, 291
 - activemq::wireformat::openwire::marshal::v1::ActiveMQMessage, 313
 - activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMarshal, 353
 - activemq::wireformat::openwire::marshal::v1::ActiveMQQueue, 389
 - activemq::wireformat::openwire::marshal::v1::ActiveMQStream, 442
 - activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueue, 464
 - activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueue, 487
 - activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueue, 511
 - activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessage, 536
 - activemq::wireformat::openwire::marshal::v1::ActiveMQTopic, 559
 - activemq::wireformat::openwire::marshal::v1::BaseCommandMarshal, 613
 - activemq::wireformat::openwire::marshal::v1::BrokerIdMarshal, 700
 - activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshal, 728
 - activemq::wireformat::openwire::marshal::v1::ConnectionControl, 1066

activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller	2264
activemq::wireformat::openwire::marshal::v1::NetworkBridgeFactory	1089
activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller	1115
activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller	1145
activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller	1181
activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller	1205
activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller	1229
activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionMarshaller	1256
activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller	1281
activemq::wireformat::openwire::marshal::v1::ResponseMarshaller	1316
activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller	1409
activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller	1438
activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller	1493
activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller	1582
activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller	1676
activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller	1736
activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller	1753
activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller	1784
activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMarshaller	1807
activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMarshaller	1834
activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller	1849
activemq::wireformat::openwire::marshal::v2::ActiveMQMapMarshaller	1892
activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller	2078
activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMarshaller	2110
activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller	2134
activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMarshaller	2164
activemq::wireformat::openwire::marshal::v2::ActiveMQTempFileMarshaller	2185
activemq::wireformat::openwire::marshal::v2::ActiveMQTempFileMarshaller	2217

activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller	523	activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller	1880
activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller	548	activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller	2062
activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller	571	activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller	2098
activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller	633	activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller	2122
activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller	712	activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller	2148
activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller	740	activemq::wireformat::openwire::marshal::v2::MessageMarshaller	2168
activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller	1078	activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller	2209
activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller	1101	activemq::wireformat::openwire::marshal::v2::NetworkBridgeMarshaller	2252
activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller	1127	activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller	2324
activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller	1153	activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller	2426
activemq::wireformat::openwire::marshal::v2::ConsumerGroupMarshaller	1189	activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller	2452
activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller	1213	activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller	2476
activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller	1241	activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller	2550
activemq::wireformat::openwire::marshal::v2::ContactCommandMarshaller	1264	activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionMarshaller	2570
activemq::wireformat::openwire::marshal::v2::DataActiveResponseMarshaller	1289	activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller	2589
activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller	1328	activemq::wireformat::openwire::marshal::v2::ResponseMarshaller	2622
activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller	1398	activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller	2692
activemq::wireformat::openwire::marshal::v2::DiscardResponseMarshaller	1422	activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller	2707
activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller	1489	activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller	2774
activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller	1578	activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller	2929
activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller	1672	activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller	3021
activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller	1724	activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller	3059
activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller	1749	activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller	3165
activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller	1772	activemq::wireformat::openwire::marshal::v2::XATransactionInfoMarshaller	3199
activemq::wireformat::openwire::marshal::v2::JournalTransactionInfoMarshaller	1795	activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMarshaller	149
activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller	1818	activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMarshaller	184
activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller	1845	activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller	252

activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller	1680	activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller
287		
activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller	1732	activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller
309		
activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller	1757	activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller
349		
activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMessageMarshaller	1780	activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller
385		
activemq::wireformat::openwire::marshal::v3::ActiveMQSequenceMessageMarshaller	1799	activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller
438		
activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller	1826	activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller
461		
activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller	1853	activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller
483		
activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller	1884	activemq::wireformat::openwire::marshal::v3::LocalTransactionMarshaller
507		
activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller	2070	activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller
532		
activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller	2106	activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller
556		
activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller	2130	activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller
606		
activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller	2156	activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller
696		
activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller	2177	activemq::wireformat::openwire::marshal::v3::MessageMarshaller
724		
activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller	2213	activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller
1062		
activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller	2256	activemq::wireformat::openwire::marshal::v3::NetworkBridgeFailureMarshaller
1086		
activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller	2329	activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller
1111		
activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller	2430	activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller
1137		
activemq::wireformat::openwire::marshal::v3::ConsumerGroupMarshaller	2456	activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller
1173		
activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller	2480	activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller
1197		
activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller	2554	activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller
1225		
activemq::wireformat::openwire::marshal::v3::ContentCommandMarshaller	2578	activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionMarshaller
1249		
activemq::wireformat::openwire::marshal::v3::DataActiveResponseMarshaller	2593	activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller
1273		
activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller	2627	activemq::wireformat::openwire::marshal::v3::ResponseMarshaller
1312		
activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller	2700	activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller
1401		
activemq::wireformat::openwire::marshal::v3::DiscardRequestMarshaller	2723	activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller
1426		
activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller	2770	activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller
1497		
activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller	2917	activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller
1586		

activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller	1252
3035	
activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller	1277
3047	
activemq::wireformat::openwire::marshal::v3::WireFormatInwardMarshaller	1324
3181	
activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller	1405
3203	
activemq::wireformat::openwire::marshal::v4::ActiveMQBlobWireFormatMarshaller	1430
157	
activemq::wireformat::openwire::marshal::v4::ActiveMQByteWireFormatMarshaller	1501
192	
activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller	1590
260	
activemq::wireformat::openwire::marshal::v4::ActiveMQMapWireFormatMarshaller	1684
295	
activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller	1728
317	
activemq::wireformat::openwire::marshal::v4::ActiveMQObjectWireFormatMarshaller	1761
357	
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueWireFormatMarshaller	1776
393	
activemq::wireformat::openwire::marshal::v4::ActiveMQSequenceWireFormatMarshaller	1803
446	
activemq::wireformat::openwire::marshal::v4::ActiveMQTempDestinationMarshaller	1830
468	
activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller	1857
491	
activemq::wireformat::openwire::marshal::v4::ActiveMQTempTopicMarshaller	1888
515	
activemq::wireformat::openwire::marshal::v4::ActiveMQTextWireFormatMarshaller	2074
540	
activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller	2102
563	
activemq::wireformat::openwire::marshal::v4::BaseCommandWireFormatMarshaller	2126
620	
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller	2152
704	
activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller	2173
732	
activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller	2225
1070	
activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller	2268
1093	
activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller	2333
1119	
activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller	2434
1141	
activemq::wireformat::openwire::marshal::v4::ConsumerControlMarshaller	2468
1177	
activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller	2488
1201	
activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller	2558
1233	
activemq::wireformat::openwire::marshal::v4::ControlCommandMarshaller	
activemq::wireformat::openwire::marshal::v4::DataArrayResponseMarshaller	
activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller	
activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller	
activemq::wireformat::openwire::marshal::v4::DiscoveryEventManager	
activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller	
activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller	
activemq::wireformat::openwire::marshal::v4::IntegerResponseMarshaller	
activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller	
activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller	
activemq::wireformat::openwire::marshal::v4::JournalTraceMarshaller	
activemq::wireformat::openwire::marshal::v4::JournalTransactionMarshaller	
activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller	
activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller	
activemq::wireformat::openwire::marshal::v4::LocalTransactionMarshaller	
activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller	
activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller	
activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller	
activemq::wireformat::openwire::marshal::v4::MessageMarshaller	
activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller	
activemq::wireformat::openwire::marshal::v4::NetworkBridgeMarshaller	
activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller	
activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller	
activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller	
activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller	
activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller	

activemq::wireformat::openwire::marshal::v4::RemoteSubscriptionInfoMarshaller	1074
2582	
activemq::wireformat::openwire::marshal::v4::ReplyCommandMarshaller	1097
2601	
activemq::wireformat::openwire::marshal::v4::ResponseMarshaller	1123
2640	
activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller	1149
2696	
activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller	1185
2715	
activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller	1209
2766	
activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller	1237
2925	
activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller	1260
3028	
activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller	1285
3055	
activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller	1320
3177	
activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller	1413
3207	
activemq::wireformat::openwire::marshal::v5::ActiveMQBlockWireFormatMarshaller	1434
161	
activemq::wireformat::openwire::marshal::v5::ActiveMQByteMessageMarshaller	1505
196	
activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller	1594
264	
activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller	1688
299	
activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller	1740
321	
activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller	1765
361	
activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller	1788
397	
activemq::wireformat::openwire::marshal::v5::ActiveMQStackingMessageMarshaller	1811
450	
activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller	1822
472	
activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller	1861
495	
activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller	1896
519	
activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller	2066
544	
activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller	2114
567	
activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller	2138
626	
activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller	2160
708	
activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller	2181
736	

- activemq::wireformat::openwire::marshal::v5::MessageIDMarshaller, 2221
- activemq::wireformat::openwire::marshal::v5::NetworkFilterMarshaller, 2260
- activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller, 2341
- activemq::wireformat::openwire::marshal::v5::ProducerIDMarshaller, 2438
- activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller, 2460
- activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller, 2484
- activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller, 2546
- activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionInfoMarshaller, 2566
- activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller, 2597
- activemq::wireformat::openwire::marshal::v5::ResponseMarshaller, 2631
- activemq::wireformat::openwire::marshal::v5::SessionIDMarshaller, 2684
- activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller, 2719
- activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller, 2778
- activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller, 2921
- activemq::wireformat::openwire::marshal::v5::TransactionIDMarshaller, 3032
- activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller, 3043
- activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller, 3169
- activemq::wireformat::openwire::marshal::v5::XATransactionIDMarshaller, 3215
- tightMarshal2 1282
- activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller, 649
- activemq::wireformat::openwire::marshal::DataStreamMarshaller, 1360
- activemq::wireformat::openwire::marshal::v1::ActiveMQBlobWireFormatMarshaller, 153
- activemq::wireformat::openwire::marshal::v1::ActiveMQByteWireFormatMarshaller, 189
- activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller, 256
- activemq::wireformat::openwire::marshal::v1::ActiveMQMapWireFormatMarshaller, 291
- activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller, 314
- activemq::wireformat::openwire::marshal::v1::ActiveMQObjectWireFormatMarshaller, 354
- activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller, 1389
- activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMarshaller, 1422
- activemq::wireformat::openwire::marshal::v1::ActiveMQTempMarshaller, 1465
- activemq::wireformat::openwire::marshal::v1::ActiveMQTempMarshaller, 1488
- activemq::wireformat::openwire::marshal::v1::ActiveMQTempMarshaller, 1511
- activemq::wireformat::openwire::marshal::v1::ActiveMQTextMarshaller, 1536
- activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller, 1560
- activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller, 1584
- activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller, 1600
- activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller, 1628
- activemq::wireformat::openwire::marshal::v1::ConnectionContainerMarshaller, 1666
- activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller, 1690
- activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller, 1715
- activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller, 1745
- activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller, 1781
- activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller, 1805
- activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller, 1829
- activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller, 1857
- activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller, 1870
- activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller, 1882
- activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller, 1910
- activemq::wireformat::openwire::marshal::v1::DiscoveryEventManagerMarshaller, 1938
- activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller, 1994
- activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller, 2022
- activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller, 2076
- activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller, 2136
- activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller, 2173
- activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller, 2201

1784	3212
activemq::wireformat::openwire::marshal::v1::JournalTransactionInfoMarshaller	activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMarshaller
1807	165
activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller	activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMarshaller
1834	201
activemq::wireformat::openwire::marshal::v1::LastBatchInfoMarshaller	activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller
1849	268
activemq::wireformat::openwire::marshal::v1::LocalTransactionInfoMarshaller	activemq::wireformat::openwire::marshal::v2::ActiveMQMapMarshaller
1893	303
activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller	activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller
2079	326
activemq::wireformat::openwire::marshal::v1::MessageDispatchInfoMarshaller	activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMarshaller
2110	366
activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller	activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller
2135	401
activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller	activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMarshaller
2164	454
activemq::wireformat::openwire::marshal::v1::MessageMarshaller	activemq::wireformat::openwire::marshal::v2::ActiveMQTemplateMarshaller
2185	476
activemq::wireformat::openwire::marshal::v1::MessagePublishInfoMarshaller	activemq::wireformat::openwire::marshal::v2::ActiveMQTemplateMarshaller
2218	500
activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller	activemq::wireformat::openwire::marshal::v2::ActiveMQTemplateMarshaller
2264	524
activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller	activemq::wireformat::openwire::marshal::v2::ActiveMQTextMarshaller
2338	548
activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller	activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller
2442	572
activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller	activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller
2464	634
activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller	activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller
2493	712
activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller	activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller
2543	740
activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller	activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller
2575	1078
activemq::wireformat::openwire::marshal::v1::ReplaceCommandMarshaller	activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller
2606	1102
activemq::wireformat::openwire::marshal::v1::ResponseMarshaller	activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller
2636	1127
activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller	activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller
2688	1153
activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller	activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller
2712	1189
activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller	activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller
2762	1213
activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller	activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller
2914	1241
activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller	activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller
3025	1265
activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller	activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller
3051	1290
activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller	activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller
3173	1329
activemq::wireformat::openwire::marshal::v1::XATransactionInfoMarshaller	activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller

1398	2692
activemq::wireformat::openwire::marshal::v2::DisconnectWireFormatHandler; openwire::marshal::v2::SessionInfoMarshaller	
1423	2708
activemq::wireformat::openwire::marshal::v2::ExceptionHandler; openwire::marshal::v2::ShutdownInfoMarshaller	
1490	2774
activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller; openwire::marshal::v2::SubscriptionInfoMarshaller	
1578	2929
activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller; openwire::marshal::v2::TransactionIdMarshaller	
1672	3021
activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller; openwire::marshal::v2::TransactionInfoMarshaller	
1724	3059
activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller; openwire::marshal::v2::WireFormatInfoMarshaller	
1749	3165
activemq::wireformat::openwire::marshal::v2::JournalTransactionInfoMarshaller; openwire::marshal::v2::XATransactionInfoMarshaller	
1772	3200
activemq::wireformat::openwire::marshal::v2::JournalTransactionInfoMarshaller; openwire::marshal::v3::ActiveMQBlobMarshaller	
1796	149
activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller; openwire::marshal::v3::ActiveMQBytesMarshaller	
1818	185
activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller; openwire::marshal::v3::ActiveMQDestinationMarshaller	
1845	253
activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller; openwire::marshal::v3::ActiveMQMapMarshaller	
1881	287
activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller; openwire::marshal::v3::ActiveMQMessageMarshaller	
2063	310
activemq::wireformat::openwire::marshal::v2::MessageDispatchInfoMarshaller; openwire::marshal::v3::ActiveMQObjectMarshaller	
2098	350
activemq::wireformat::openwire::marshal::v2::MessageDispatchInfoMarshaller; openwire::marshal::v3::ActiveMQQueueMarshaller	
2123	385
activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller; openwire::marshal::v3::ActiveMQStreamMarshaller	
2149	438
activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller; openwire::marshal::v3::ActiveMQTemplateMarshaller	
2169	461
activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller; openwire::marshal::v3::ActiveMQTemplateMarshaller	
2210	484
activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller; openwire::marshal::v3::ActiveMQTemplateMarshaller	
2252	508
activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller; openwire::marshal::v3::ActiveMQTextMarshaller	
2325	532
activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller; openwire::marshal::v3::ActiveMQTopicMarshaller	
2426	556
activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller; openwire::marshal::v3::BaseCommandMarshaller	
2453	607
activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller; openwire::marshal::v3::BrokerIdMarshaller	
2477	697
activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller; openwire::marshal::v3::BrokerInfoMarshaller	
2551	724
activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller; openwire::marshal::v3::ConnectionContainerMarshaller	
2571	1062
activemq::wireformat::openwire::marshal::v2::ReplyCommandMarshaller; openwire::marshal::v3::ConnectionErrorMarshaller	
2590	1086
activemq::wireformat::openwire::marshal::v2::ResponseMarshaller; openwire::marshal::v3::ConnectionIdMarshaller	
2623	1112
activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller; openwire::marshal::v3::ConnectionInfoMarshaller	

1137	2430
activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller	activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller
1173	2456
activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller	activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller
1197	2481
activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller	activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller
1225	2555
activemq::wireformat::openwire::marshal::v3::ConsumerQueueIdMarshaller	activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionMarshaller
1249	2579
activemq::wireformat::openwire::marshal::v3::DataAndResponseMarshaller	activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller
1274	2594
activemq::wireformat::openwire::marshal::v3::DataAndResponseMarshaller	activemq::wireformat::openwire::marshal::v3::ResponseMarshaller
1313	2627
activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller	activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller
1402	2700
activemq::wireformat::openwire::marshal::v3::DiscoveryInfoMarshaller	activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller
1426	2724
activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller	activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller
1498	2770
activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller	activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller
1586	2918
activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller	activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller
1680	3036
activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller	activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller
1732	3047
activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller	activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller
1757	3181
activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller	activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller
1780	3204
activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller	activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMarshaller
1800	157
activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller	activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMarshaller
1826	193
activemq::wireformat::openwire::marshal::v3::LastBatchCompInfoMarshaller	activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller
1853	260
activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller	activemq::wireformat::openwire::marshal::v4::ActiveMQMapMarshaller
1885	295
activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller	activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller
2071	318
activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller	activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMarshaller
2106	358
activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller	activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller
2131	393
activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller	activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMarshaller
2156	446
activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller	activemq::wireformat::openwire::marshal::v4::ActiveMQTempMarshaller
2177	468
activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller	activemq::wireformat::openwire::marshal::v4::ActiveMQTempMarshaller
2214	492
activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller	activemq::wireformat::openwire::marshal::v4::ActiveMQTempMarshaller
2256	516
activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller	activemq::wireformat::openwire::marshal::v4::ActiveMQTextMarshaller
2329	540
activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller	activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMarshaller

564	2102
activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller	activemq::wireformat::openwire::marshal::v4::MessageDispatcher
621	2127
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller	activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller
704	2153
activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller	activemq::wireformat::openwire::marshal::v4::MessageMarshaller
732	2173
activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller	activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller
1070	2226
activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller	activemq::wireformat::openwire::marshal::v4::NetworkBridgeFilter
1094	2268
activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller	activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller
1119	2333
activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller	activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller
1141	2434
activemq::wireformat::openwire::marshal::v4::ConsumerGroupMarshaller	activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller
1177	2468
activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller	activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller
1201	2489
activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller	activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller
1233	2559
activemq::wireformat::openwire::marshal::v4::ContactCommandMarshaller	activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionMarshaller
1253	2583
activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller	activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller
1278	2602
activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller	activemq::wireformat::openwire::marshal::v4::ResponseMarshaller
1325	2641
activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller	activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller
1406	2696
activemq::wireformat::openwire::marshal::v4::DiscoveryResponseMarshaller	activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller
1430	2716
activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller	activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller
1502	2766
activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller	activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller
1590	2925
activemq::wireformat::openwire::marshal::v4::IntegrationResponseMarshaller	activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller
1684	3028
activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller	activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller
1728	3055
activemq::wireformat::openwire::marshal::v4::JournalTopicAckMarshaller	activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller
1761	3177
activemq::wireformat::openwire::marshal::v4::JournalTransactionInfoMarshaller	activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller
1776	3208
activemq::wireformat::openwire::marshal::v4::JournalTransactionInfoMarshaller	activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMarshaller
1803	161
activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller	activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMarshaller
1830	197
activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller	activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller
1857	264
activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller	activemq::wireformat::openwire::marshal::v5::ActiveMQMapMarshaller
1889	299
activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller	activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller
2075	322
activemq::wireformat::openwire::marshal::v4::MessageDispatcherMarshaller	activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMarshaller

362	1765
activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMessageMarshaller	activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller
397	1788
activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMessageMarshaller	activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller
450	1811
activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMessageMarshaller	activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller
472	1822
activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMessageMarshaller	activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller
496	1861
activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMessageMarshaller	activemq::wireformat::openwire::marshal::v5::LocalTransactionMarshaller
520	1897
activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMessageMarshaller	activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller
544	2067
activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMessageMarshaller	activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller
568	2114
activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller	activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller
627	2139
activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller	activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller
708	2160
activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller	activemq::wireformat::openwire::marshal::v5::MessageMarshaller
736	2181
activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller	activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller
1074	2222
activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller	activemq::wireformat::openwire::marshal::v5::NetworkBridgeMarshaller
1098	2260
activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller	activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller
1123	2342
activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller	activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller
1149	2438
activemq::wireformat::openwire::marshal::v5::ConsumerGroupMarshaller	activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller
1185	2460
activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller	activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller
1209	2485
activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller	activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller
1237	2547
activemq::wireformat::openwire::marshal::v5::ContentCommandMarshaller	activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionMarshaller
1261	2567
activemq::wireformat::openwire::marshal::v5::DataActiveResponseMarshaller	activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller
1286	2598
activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller	activemq::wireformat::openwire::marshal::v5::ResponseMarshaller
1321	2632
activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller	activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller
1414	2684
activemq::wireformat::openwire::marshal::v5::DiscoveryInfoMarshaller	activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller
1434	2720
activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller	activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller
1506	2778
activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller	activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshaller
1594	2922
activemq::wireformat::openwire::marshal::v5::IntegrationResponseMarshaller	activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller
1688	3032
activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller	activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller
1740	3043
activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller	activemq::wireformat::openwire::marshal::v5::WireFormatInfoMarshaller

3169	activemq::wireformat::openwire::marshal::v1::ActiveMQMapM
activemq::wireformat::openwire::marshal::v5::XATrans	202
3216	activemq::wireformat::openwire::marshal::v1::ActiveMQMessa
tightMarshalBrokerError1	314
activemq::wireformat::openwire::marshal::BaseDataStreamM	314
649	354
tightMarshalBrokerError2	activemq::wireformat::openwire::marshal::v1::ActiveMQQueue
activemq::wireformat::openwire::marshal::BaseDataStreamM	390
650	activemq::wireformat::openwire::marshal::v1::ActiveMQStream
tightMarshalCachedObject1	443
activemq::wireformat::openwire::marshal::BaseDataStreamM	443
650	465
tightMarshalCachedObject2	activemq::wireformat::openwire::marshal::v1::ActiveMQTemp
activemq::wireformat::openwire::marshal::BaseDataStreamM	488
650	activemq::wireformat::openwire::marshal::v1::ActiveMQTemp
tightMarshalLong1	512
activemq::wireformat::openwire::marshal::BaseDataStreamM	512
651	537
tightMarshalLong2	activemq::wireformat::openwire::marshal::v1::ActiveMQTopic
activemq::wireformat::openwire::marshal::BaseDataStreamM	560
651	activemq::wireformat::openwire::marshal::v1::BaseCommandM
tightMarshalNestedObject1	615
activemq::wireformat::openwire::marshal::BaseDataStreamM	615
652	701
activemq::wireformat::openwire::OpenWireFormat	activemq::wireformat::openwire::marshal::v1::BrokerInfoMarsh
2306	729
tightMarshalNestedObject2	activemq::wireformat::openwire::marshal::v1::ConnectionCont
activemq::wireformat::openwire::marshal::BaseDataStreamM	1067
652	activemq::wireformat::openwire::marshal::v1::ConnectionError
activemq::wireformat::openwire::OpenWireFormat	1090
2307	activemq::wireformat::openwire::marshal::v1::ConnectionIdMa
tightMarshalObjectArray1	1116
activemq::wireformat::openwire::marshal::BaseDataStreamM	1116
652	1146
tightMarshalObjectArray2	activemq::wireformat::openwire::marshal::v1::ConsumerContro
activemq::wireformat::openwire::marshal::BaseDataStreamM	1182
653	activemq::wireformat::openwire::marshal::v1::ConsumerIdMar
tightMarshalString1	1206
activemq::wireformat::openwire::marshal::BaseDataStreamM	1206
653	1230
tightMarshalString2	activemq::wireformat::openwire::marshal::v1::ControlComman
activemq::wireformat::openwire::marshal::BaseDataStreamM	1257
654	activemq::wireformat::openwire::marshal::v1::DataArrayRespo
tightUnmarshal	1282
activemq::wireformat::openwire::marshal::BaseDataStreamM	1282
654	1317
activemq::wireformat::openwire::marshal::DataStreamM	1317
1366	1410
activemq::wireformat::openwire::marshal::v1::ActiveMQBlkWireFormat	1410
154	1439
activemq::wireformat::openwire::marshal::v1::ActiveMQByteWireFormat	1439
189	1494
activemq::wireformat::openwire::marshal::v1::ActiveMQDestinat	1494
257	1583

activemq::wireformat::openwire::marshal::v1::IntegerResponseInfoMarshaler	3025	openwire::marshal::v1::TransactionIdMarshaler
1677		
activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaler	3052	openwire::marshal::v1::TransactionInfoMarshaler
1737		
activemq::wireformat::openwire::marshal::v1::JournalTopicAckInfoMarshaler	3174	openwire::marshal::v1::WireFormatInfoMarshaler
1754		
activemq::wireformat::openwire::marshal::v1::JournalTransactionInfoMarshaler	3212	openwire::marshal::v1::XATransactionIdMarshaler
1785		
activemq::wireformat::openwire::marshal::v1::JournalTransactionInfoMarshaler	166	openwire::marshal::v2::ActiveMQBlobMarshaler
1808		
activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaler	201	openwire::marshal::v2::ActiveMQBytesMarshaler
1835		
activemq::wireformat::openwire::marshal::v1::LastPartialCommandInfoMarshaler	269	openwire::marshal::v2::ActiveMQDestinationMarshaler
1850		
activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaler	304	openwire::marshal::v2::ActiveMQMapMarshaler
1893		
activemq::wireformat::openwire::marshal::v1::MessageAckInfoMarshaler	326	openwire::marshal::v2::ActiveMQMessageMarshaler
2079		
activemq::wireformat::openwire::marshal::v1::MessageDispatchInfoMarshaler	366	openwire::marshal::v2::ActiveMQObjectMarshaler
2111		
activemq::wireformat::openwire::marshal::v1::MessageDispatchInfoNotificationMarshaler	402	openwire::marshal::v2::ActiveMQQueueMarshaler
2135		
activemq::wireformat::openwire::marshal::v1::MessageIdInfoMarshaler	455	openwire::marshal::v2::ActiveMQStreamMarshaler
2165		
activemq::wireformat::openwire::marshal::v1::MessageIdInfoMarshaler	476	openwire::marshal::v2::ActiveMQTemplateMarshaler
2186		
activemq::wireformat::openwire::marshal::v1::MessagePublishInfoMarshaler	500	openwire::marshal::v2::ActiveMQTempQueueMarshaler
2218		
activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaler	524	openwire::marshal::v2::ActiveMQTempQueueMarshaler
2265		
activemq::wireformat::openwire::marshal::v1::PartialCommandInfoMarshaler	549	openwire::marshal::v2::ActiveMQTextMarshaler
2338		
activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaler	572	openwire::marshal::v2::ActiveMQTopicMarshaler
2443		
activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaler	635	openwire::marshal::v2::BaseCommandMarshaler
2465		
activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaler	713	openwire::marshal::v2::BrokerIdMarshaler
2493		
activemq::wireformat::openwire::marshal::v1::RemoteSubscriptionInfoMarshaler	741	openwire::marshal::v2::BrokerInfoMarshaler
2543		
activemq::wireformat::openwire::marshal::v1::RemoteSubscriptionInfoMarshaler	1079	openwire::marshal::v2::ConnectionContextMarshaler
2575		
activemq::wireformat::openwire::marshal::v1::ReplyCommandInfoMarshaler	1102	openwire::marshal::v2::ConnectionErrorMarshaler
2606		
activemq::wireformat::openwire::marshal::v1::ResponseInfoMarshaler	1128	openwire::marshal::v2::ConnectionIdMarshaler
2637		
activemq::wireformat::openwire::marshal::v1::SessionIdInfoMarshaler	1154	openwire::marshal::v2::ConnectionInfoMarshaler
2689		
activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaler	1190	openwire::marshal::v2::ConsumerControlMarshaler
2712		
activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaler	1214	openwire::marshal::v2::ConsumerIdMarshaler
2763		
activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaler	1242	openwire::marshal::v2::ConsumerInfoMarshaler
2914		

activemq::wireformat::openwire::marshal::v2::CommandMarshaller	2571	activemq::wireformat::openwire::marshal::v2::RemoveSubscription	2571
1265			
activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller	2590	activemq::wireformat::openwire::marshal::v2::ReplayCommand	2590
1290			
activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller	2623	activemq::wireformat::openwire::marshal::v2::ResponseMarshaller	2623
1329			
activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller	2693	activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller	2693
1398			
activemq::wireformat::openwire::marshal::v2::DiscoveryInfoMarshaller	2708	activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller	2708
1423			
activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller	2775	activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller	2775
1490			
activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller	2930	activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller	2930
1579			
activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller	3022	activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller	3022
1673			
activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller	3060	activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller	3060
1725			
activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller	3166	activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller	3166
1750			
activemq::wireformat::openwire::marshal::v2::JournalTransactionInfoMarshaller	3200	activemq::wireformat::openwire::marshal::v2::XATransactionInfoMarshaller	3200
1773			
activemq::wireformat::openwire::marshal::v2::JournalTransactionInfoMarshaller	150	activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMarshaller	150
1796			
activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller	185	activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMarshaller	185
1819			
activemq::wireformat::openwire::marshal::v2::LastReceivedCommandMarshaller	253	activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller	253
1846			
activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller	288	activemq::wireformat::openwire::marshal::v3::ActiveMQMapMarshaller	288
1881			
activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller	310	activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller	310
2063			
activemq::wireformat::openwire::marshal::v2::MessageDispatchInfoMarshaller	350	activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMarshaller	350
2099			
activemq::wireformat::openwire::marshal::v2::MessageDispatchInfoMarshaller	386	activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller	386
2123			
activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller	439	activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMarshaller	439
2149			
activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller	462	activemq::wireformat::openwire::marshal::v3::ActiveMQTempMarshaller	462
2170			
activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller	484	activemq::wireformat::openwire::marshal::v3::ActiveMQTempMarshaller	484
2210			
activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller	508	activemq::wireformat::openwire::marshal::v3::ActiveMQTempMarshaller	508
2253			
activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller	533	activemq::wireformat::openwire::marshal::v3::ActiveMQTextMarshaller	533
2325			
activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller	556	activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller	556
2427			
activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller	609	activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller	609
2453			
activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller	697	activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller	697
2477			
activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller	725	activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller	725
2551			

activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller	2214
activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller	2257
activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller	2330
activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller	2431
activemq::wireformat::openwire::marshal::v3::ConsumerGroupViewMarshaller	2457
activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller	2481
activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller	2555
activemq::wireformat::openwire::marshal::v3::ContainerCommandMarshaller	2579
activemq::wireformat::openwire::marshal::v3::DataActiveResponseForMarshaller	2594
activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller	2628
activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller	2701
activemq::wireformat::openwire::marshal::v3::DisconnectEventMarshaller	2724
activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller	2771
activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller	2918
activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller	3036
activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller	3048
activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller	3182
activemq::wireformat::openwire::marshal::v3::JournalTransactionIdMarshaller	3204
activemq::wireformat::openwire::marshal::v3::JournalTransactionIdMarshaller	158
activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller	193
activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller	261
activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller	296
activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller	318
activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller	358
activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller	394
activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller	447
activemq::wireformat::openwire::marshal::v3::MessageMarshaller	469
activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller	
activemq::wireformat::openwire::marshal::v3::NetworkBridgeFactory	
activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller	
activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller	
activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller	
activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller	
activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller	
activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionMarshaller	
activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller	
activemq::wireformat::openwire::marshal::v3::ResponseMarshaller	
activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller	
activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller	
activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller	
activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller	
activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller	
activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller	
activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller	
activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller	
activemq::wireformat::openwire::marshal::v4::ActiveMQBlobMarshaller	
activemq::wireformat::openwire::marshal::v4::ActiveMQBytesMarshaller	
activemq::wireformat::openwire::marshal::v4::ActiveMQDestinationMarshaller	
activemq::wireformat::openwire::marshal::v4::ActiveMQMapMarshaller	
activemq::wireformat::openwire::marshal::v4::ActiveMQMessageMarshaller	
activemq::wireformat::openwire::marshal::v4::ActiveMQObjectMarshaller	
activemq::wireformat::openwire::marshal::v4::ActiveMQQueueMarshaller	
activemq::wireformat::openwire::marshal::v4::ActiveMQStreamMarshaller	
activemq::wireformat::openwire::marshal::v4::ActiveMQTempMarshaller	

activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller	492	activemq::wireformat::openwire::marshal::v4::LastPartialCommandMarshaller	1858
activemq::wireformat::openwire::marshal::v4::ActiveMQTempQueueMarshaller	516	activemq::wireformat::openwire::marshal::v4::LocalTransactionIdMarshaller	1889
activemq::wireformat::openwire::marshal::v4::ActiveMQTextMessageMarshaller	541	activemq::wireformat::openwire::marshal::v4::MessageAckMarshaller	2075
activemq::wireformat::openwire::marshal::v4::ActiveMQTopicMessageMarshaller	564	activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller	2103
activemq::wireformat::openwire::marshal::v4::BaseCommandMarshaller	622	activemq::wireformat::openwire::marshal::v4::MessageDispatchMarshaller	2127
activemq::wireformat::openwire::marshal::v4::BrokerIdMarshaller	705	activemq::wireformat::openwire::marshal::v4::MessageIdMarshaller	2153
activemq::wireformat::openwire::marshal::v4::BrokerInfoMarshaller	733	activemq::wireformat::openwire::marshal::v4::MessageMarshaller	2174
activemq::wireformat::openwire::marshal::v4::ConnectionControlMarshaller	1071	activemq::wireformat::openwire::marshal::v4::MessagePullMarshaller	2226
activemq::wireformat::openwire::marshal::v4::ConnectionErrorMarshaller	1094	activemq::wireformat::openwire::marshal::v4::NetworkBridgeFailureMarshaller	2269
activemq::wireformat::openwire::marshal::v4::ConnectionIdMarshaller	1120	activemq::wireformat::openwire::marshal::v4::PartialCommandMarshaller	2334
activemq::wireformat::openwire::marshal::v4::ConnectionInfoMarshaller	1142	activemq::wireformat::openwire::marshal::v4::ProducerAckMarshaller	2435
activemq::wireformat::openwire::marshal::v4::ConsumerGroupMarshaller	1178	activemq::wireformat::openwire::marshal::v4::ProducerIdMarshaller	2469
activemq::wireformat::openwire::marshal::v4::ConsumerIdMarshaller	1202	activemq::wireformat::openwire::marshal::v4::ProducerInfoMarshaller	2489
activemq::wireformat::openwire::marshal::v4::ConsumerInfoMarshaller	1234	activemq::wireformat::openwire::marshal::v4::RemoveInfoMarshaller	2559
activemq::wireformat::openwire::marshal::v4::ContactCommandMarshaller	1253	activemq::wireformat::openwire::marshal::v4::RemoveSubscriptionMarshaller	2583
activemq::wireformat::openwire::marshal::v4::DataActiveResponseMarshaller	1278	activemq::wireformat::openwire::marshal::v4::ReplayCommandMarshaller	2602
activemq::wireformat::openwire::marshal::v4::DataResponseMarshaller	1325	activemq::wireformat::openwire::marshal::v4::ResponseMarshaller	2641
activemq::wireformat::openwire::marshal::v4::DestinationInfoMarshaller	1406	activemq::wireformat::openwire::marshal::v4::SessionIdMarshaller	2697
activemq::wireformat::openwire::marshal::v4::DiscardResponseMarshaller	1431	activemq::wireformat::openwire::marshal::v4::SessionInfoMarshaller	2716
activemq::wireformat::openwire::marshal::v4::ExceptionResponseMarshaller	1502	activemq::wireformat::openwire::marshal::v4::ShutdownInfoMarshaller	2767
activemq::wireformat::openwire::marshal::v4::FlushCommandMarshaller	1591	activemq::wireformat::openwire::marshal::v4::SubscriptionInfoMarshaller	2926
activemq::wireformat::openwire::marshal::v4::IntegratedResponseMarshaller	1685	activemq::wireformat::openwire::marshal::v4::TransactionIdMarshaller	3029
activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller	1729	activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller	3056
activemq::wireformat::openwire::marshal::v4::JournalQueueAckMarshaller	1762	activemq::wireformat::openwire::marshal::v4::WireFormatInfoMarshaller	3178
activemq::wireformat::openwire::marshal::v4::JournalQueueMarshaller	1777	activemq::wireformat::openwire::marshal::v4::XATransactionIdMarshaller	3208
activemq::wireformat::openwire::marshal::v4::JournalQueueMarshaller	1804	activemq::wireformat::openwire::marshal::v5::ActiveMQBlobMarshaller	162
activemq::wireformat::openwire::marshal::v4::KeepAliveInfoMarshaller	1831	activemq::wireformat::openwire::marshal::v5::ActiveMQBytesMarshaller	197

activemq::wireformat::openwire::marshal::v5::ActiveMQDestinationMarshaller	1595	activemq::wireformat::openwire::marshal::v5::FlushCommandMarshaller	1595
265		activemq::wireformat::openwire::marshal::v5::IntegerResponseMarshaller	1689
activemq::wireformat::openwire::marshal::v5::ActiveMQMapMessageMarshaller	1689	activemq::wireformat::openwire::marshal::v5::JournalQueueAckMarshaller	1741
300		activemq::wireformat::openwire::marshal::v5::JournalTopicAckMarshaller	1766
activemq::wireformat::openwire::marshal::v5::ActiveMQMessageMarshaller	1741	activemq::wireformat::openwire::marshal::v5::JournalTraceMarshaller	1789
322		activemq::wireformat::openwire::marshal::v5::JournalTransactionMarshaller	1812
activemq::wireformat::openwire::marshal::v5::ActiveMQObjectMessageMarshaller	1766	activemq::wireformat::openwire::marshal::v5::KeepAliveInfoMarshaller	1823
362		activemq::wireformat::openwire::marshal::v5::LastPartialCommandMarshaller	1862
activemq::wireformat::openwire::marshal::v5::ActiveMQQueueMarshaller	1789	activemq::wireformat::openwire::marshal::v5::LocalTransactionMarshaller	1897
398		activemq::wireformat::openwire::marshal::v5::MessageAckMarshaller	2067
activemq::wireformat::openwire::marshal::v5::ActiveMQQueueViewMessageMarshaller	1812	activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller	2115
451		activemq::wireformat::openwire::marshal::v5::MessageDispatchMarshaller	2139
activemq::wireformat::openwire::marshal::v5::ActiveMQTempDestinationMarshaller	1823	activemq::wireformat::openwire::marshal::v5::MessageIdMarshaller	2161
473		activemq::wireformat::openwire::marshal::v5::MessageMarshaller	2182
activemq::wireformat::openwire::marshal::v5::ActiveMQTempQueueMarshaller	1862	activemq::wireformat::openwire::marshal::v5::MessagePullMarshaller	2222
496		activemq::wireformat::openwire::marshal::v5::NetworkBridgeForwardMarshaller	2261
activemq::wireformat::openwire::marshal::v5::ActiveMQTempTopicMarshaller	1897	activemq::wireformat::openwire::marshal::v5::PartialCommandMarshaller	2342
520		activemq::wireformat::openwire::marshal::v5::ProducerAckMarshaller	2439
activemq::wireformat::openwire::marshal::v5::ActiveMQTextMessageMarshaller	2067	activemq::wireformat::openwire::marshal::v5::ProducerIdMarshaller	2461
545		activemq::wireformat::openwire::marshal::v5::ProducerInfoMarshaller	2485
activemq::wireformat::openwire::marshal::v5::ActiveMQTopicMarshaller	2115	activemq::wireformat::openwire::marshal::v5::RemoveInfoMarshaller	2547
568		activemq::wireformat::openwire::marshal::v5::RemoveSubscriptionMarshaller	2567
activemq::wireformat::openwire::marshal::v5::BaseCommandMarshaller	2139	activemq::wireformat::openwire::marshal::v5::ReplayCommandMarshaller	2598
629		activemq::wireformat::openwire::marshal::v5::ResponseMarshaller	2632
activemq::wireformat::openwire::marshal::v5::BrokerIdMarshaller	2161	activemq::wireformat::openwire::marshal::v5::SessionIdMarshaller	2685
709		activemq::wireformat::openwire::marshal::v5::SessionInfoMarshaller	2720
activemq::wireformat::openwire::marshal::v5::BrokerInfoMarshaller	2182	activemq::wireformat::openwire::marshal::v5::ShutdownInfoMarshaller	2779
737			
activemq::wireformat::openwire::marshal::v5::ConnectionControlMarshaller	2222		
1075			
activemq::wireformat::openwire::marshal::v5::ConnectionErrorMarshaller	2261		
1098			
activemq::wireformat::openwire::marshal::v5::ConnectionIdMarshaller	2342		
1124			
activemq::wireformat::openwire::marshal::v5::ConnectionInfoMarshaller	2439		
1150			
activemq::wireformat::openwire::marshal::v5::ConsumerGroupViewMarshaller	2461		
1186			
activemq::wireformat::openwire::marshal::v5::ConsumerIdMarshaller	2485		
1210			
activemq::wireformat::openwire::marshal::v5::ConsumerInfoMarshaller	2547		
1238			
activemq::wireformat::openwire::marshal::v5::ConsumerQueueViewMarshaller	2567		
1261			
activemq::wireformat::openwire::marshal::v5::DataActiveResponseMarshaller	2598		
1286			
activemq::wireformat::openwire::marshal::v5::DataResponseMarshaller	2632		
1321			
activemq::wireformat::openwire::marshal::v5::DestinationInfoMarshaller	2685		
1414			
activemq::wireformat::openwire::marshal::v5::DiscardQueueViewMarshaller	2720		
1435			
activemq::wireformat::openwire::marshal::v5::ExceptionResponseMarshaller	2779		
1506			

- activemq::wireformat::openwire::marshal::v5::SubscriptionInfoMarshallerMessage, 2035
- 2922
- decaf::util::UUID, 3146
- activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller, 3033
- decaf::util::concurrent::TimeUnit, 3007
- activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller, 3044
- activemq::util::ActiveMQProperties, 378
- activemq::wireformat::openwire::marshal::v5::WireFormatGMSBMPmarshaller, 3170
- decaf::util::AbstractCollection, 129
- activemq::wireformat::openwire::marshal::v5::XATransactionIDMarshaller, 3216
- decaf::util::concurrent::SynchronousQueue, 2953
- tightUnmarshalBrokerError
- activemq::wireformat::openwire::marshal::BaseDataStreamMarshallerProperties, 2502
- 655
- decaf::util::StlQueue, 2863
- tightUnmarshalByteArray
- decaf::util::StringTokenizer, 2906
- activemq::wireformat::openwire::marshal::BaseBinaryStringMarshaller, 655
- decaf::lang::Integer, 1663
- tightUnmarshalCachedObject
- decaf::lang::Long, 1945
- activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller, 655
- decaf::io::ByteArrayOutputStream, 840
- tightUnmarshalConstByteArray
- toByteArrayRef
- activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller, 656
- decaf::io::ByteArrayOutputStream, 840
- toDays
- decaf::util::concurrent::TimeUnit, 3010
- tightUnmarshalLong
- activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller, 656
- decaf::lang::Math, 2014
- tightUnmarshalNestedObject
- toDestinationOption
- activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller, 657
- activemq::wireformat::openwire::OpenWireFormatMarshaller, 2307
- activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller, 657
- toHexFromBytes
- tightUnmarshalString
- toHexString
- activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller, 657
- decaf::lang::Double, 1449
- decaf::lang::Float, 1552
- decaf::lang::Integer, 1664
- decaf::lang::Long, 1945
- timedJoin
- decaf::util::concurrent::TimeUnit, 3009
- timedWait
- decaf::util::concurrent::TimeUnit, 3010
- timeout
- activemq::commands::DestinationInfo, 1395
- activemq::commands::MessagePull, 2207
- TimeoutException
- decaf::util::concurrent::TimeoutException, 2987, 2988
- Timer
- decaf::util::Timer, 2992
- decaf::util::TimerTask, 3002
- TimerImpl
- decaf::util::TimerTask, 3002
- TimerTask
- decaf::util::TimerTask, 3001
- TimerTaskHeap
- decaf::internal::util::TimerTaskHeap, 3003
- timestamp
- TOPIC
- cms::Destination, 1388
- TOPIC_PREFIX
- activemq::wireformat::stomp::StompCommandConstants, 2873
- TOPIC_QUALIFIED_PREFIX

- activemq::commands::ActiveMQDestination, 250
- toRadians
 - decaf::lang::Math, 2014
- toSeconds
 - decaf::util::concurrent::TimeUnit, 3012
- toStream
 - activemq::wireformat::stomp::StompFrame, 2878
- toString
 - activemq::commands::ActiveMQBlobMessage, 146
 - activemq::commands::ActiveMQBytesMessage, 177
 - activemq::commands::ActiveMQDestination, 248
 - activemq::commands::ActiveMQMapMessage, 284
 - activemq::commands::ActiveMQMessage, 306
 - activemq::commands::ActiveMQObjectMessage, 346
 - activemq::commands::ActiveMQQueue, 382
 - activemq::commands::ActiveMQStreamMessage, 431
 - activemq::commands::ActiveMQTempDestination, 458
 - activemq::commands::ActiveMQTempQueue, 480
 - activemq::commands::ActiveMQTempTopic, 504
 - activemq::commands::ActiveMQTextMessage, 529
 - activemq::commands::ActiveMQTopic, 553
 - activemq::commands::BaseCommand, 602
 - activemq::commands::BaseDataStructure, 662
 - activemq::commands::BooleanExpression, 680
 - activemq::commands::BrokerId, 693
 - activemq::commands::BrokerInfo, 719
 - activemq::commands::Command, 995
 - activemq::commands::ConnectionControl, 1058
 - activemq::commands::ConnectionError, 1082
 - activemq::commands::ConnectionId, 1108
 - activemq::commands::ConnectionInfo, 1133
 - activemq::commands::ConsumerControl, 1169
 - activemq::commands::ConsumerId, 1194
 - activemq::commands::ConsumerInfo, 1220
 - activemq::commands::ControlCommand, 1245
 - activemq::commands::DataArrayResponse, 1270
 - activemq::commands::DataResponse, 1309
 - activemq::commands::DataStructure, 1376
 - activemq::commands::DestinationInfo, 1394
 - activemq::commands::DiscoveryEvent, 1419
 - activemq::commands::ExceptionResponse, 1486
 - activemq::commands::FlushCommand, 1575
 - activemq::commands::IntegerResponse, 1669
 - activemq::commands::JournalQueueAck, 1721
 - activemq::commands::JournalTopicAck, 1745
 - activemq::commands::JournalTrace, 1769
 - activemq::commands::JournalTransaction, 1792
 - activemq::commands::KeepAliveInfo, 1815
 - activemq::commands::LastPartialCommand, 1842
 - activemq::commands::LocalTransactionId, 1877
 - activemq::commands::Message, 2033
 - activemq::commands::MessageAck, 2059
 - activemq::commands::MessageDispatch, 2087
 - activemq::commands::MessageDispatchNotification, 2119
 - activemq::commands::MessageId, 2145
 - activemq::commands::MessagePull, 2206
 - activemq::commands::NetworkBridgeFilter, 2249
 - activemq::commands::PartialCommand, 2321
 - activemq::commands::ProducerAck, 2423
 - activemq::commands::ProducerId, 2449
 - activemq::commands::ProducerInfo, 2473
 - activemq::commands::RemoveInfo, 2539
 - activemq::commands::RemoveSubscriptionInfo, 2563
 - activemq::commands::ReplayCommand, 2586
 - activemq::commands::Response, 2614
 - activemq::commands::SessionId, 2681
 - activemq::commands::SessionInfo, 2704
 - activemq::commands::ShutdownInfo, 2759
 - activemq::commands::SubscriptionInfo, 2910

- activemq::commands::TransactionId, 3018
- activemq::commands::TransactionInfo, 3040
- activemq::commands::WireFormatInfo, 3162
- activemq::commands::XATransactionId, 3196
- activemq::core::ActiveMQConstants, 230
- activemq::state::ConnectionState, 1160
- activemq::state::ConsumerState, 1243
- activemq::state::ProducerState, 2494
- activemq::state::SessionState, 2728
- activemq::state::TransactionState, 3061
- activemq::util::ActiveMQProperties, 378
- activemq::util::PrimitiveList, 2380
- activemq::util::PrimitiveMap, 2390
- activemq::util::PrimitiveValueNode, 2411
- activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller, 658
- cms::CMSProperties, 968
- decaf::io::ByteArrayOutputStream, 840
- decaf::lang::Boolean, 677, 678
- decaf::lang::Byte, 783
- decaf::lang::Character, 921
- decaf::lang::CharSequence, 948
- decaf::lang::Double, 1450
- decaf::lang::Float, 1553
- decaf::lang::Integer, 1664, 1665
- decaf::lang::Long, 1946
- decaf::lang::Short, 2736
- decaf::net::URI, 3107
- decaf::nio::ByteBuffer, 873
- decaf::nio::CharBuffer, 945
- decaf::nio::DoubleBuffer, 1470
- decaf::nio::FloatBuffer, 1572
- decaf::nio::IntBuffer, 1651
- decaf::nio::LongBuffer, 1965
- decaf::nio::ShortBuffer, 2756
- decaf::security::cert::Certificate, 903
- decaf::security_-
 - provider::unix::openssl::OpenSSLX500Principal, 2290
- decaf::security_-
 - provider::unix::openssl::OpenSSLX509Certificate, 2295
- decaf::util::concurrent::atomic::AtomicBoolean, 581
- decaf::util::concurrent::atomic::AtomicInteger, 586
- decaf::util::concurrent::atomic::AtomicReference, 591
- decaf::util::concurrent::locks::ReentrantLock, 2531
- decaf::util::concurrent::Semaphore, 2657
- decaf::util::concurrent::TimeUnit, 3013
- decaf::util::Date, 1381
- decaf::util::Properties, 2502
- decaf::util::UUID, 3147
- toURI
 - activemq::util::CompositeData, 1015
- toURIOption
 - activemq::core::ActiveMQConstants, 230
- toURL
 - decaf::net::URI, 3107
- track
 - activemq::state::ConnectionStateTracker, 1165
- trackBack
 - activemq::state::ConnectionStateTracker, 1165
- Tracked
 - activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller, 3015
- TRANSACTION_STATE_BEGIN
 - activemq::core::ActiveMQConstants, 229
- TRANSACTION_STATE_-
 - COMMITONEPHASE
 - activemq::core::ActiveMQConstants, 229
- TRANSACTION_STATE_-
 - COMMITTWOPHASE
 - activemq::core::ActiveMQConstants, 229
- TRANSACTION_STATE_END
 - activemq::core::ActiveMQConstants, 229
- TRANSACTION_STATE_FORGET
 - activemq::core::ActiveMQConstants, 229
- TRANSACTION_STATE_PREPARE
 - activemq::core::ActiveMQConstants, 229
- TRANSACTION_STATE_RECOVER
 - activemq::core::ActiveMQConstants, 229
- TRANSACTION_STATE_ROLLBACK
 - activemq::core::ActiveMQConstants, 229
- TransactionId
 - activemq::commands::TransactionId, 3017
- transactionId
 - activemq::commands::JournalTopicAck, 1746
 - activemq::commands::JournalTransaction, 1793
 - activemq::commands::Message, 2035
 - activemq::commands::MessageAck, 2060
 - activemq::commands::TransactionInfo, 3040
- activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller, 3023
- activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller, 3020
- activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller, 3034

- | | |
|---|--|
| activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller, | 2657- |
| 3027 | 2659 |
| activemq::wireformat::openwire::marshal::v5::TransactionIdMarshaller, | |
| 3031 | activemq::core::MessageDispatchChannel, |
| TransactionInfo | 2093 |
| activemq::commands::TransactionInfo, | decaf::internal::io::StandardErrorOutputStream, |
| 3038 | 2815 |
| TransactionInfoMarshaller | decaf::internal::io::StandardInputStream, |
| activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller, | 2823 |
| 3050 | decaf::internal::io::StandardOutputStream, |
| activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller, | 2827 |
| 3058 | decaf::internal::util::concurrent::SynchronizableImpl, |
| activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller, | 2943 |
| 3046 | decaf::io::BlockingByteArrayInputStream, |
| activemq::wireformat::openwire::marshal::v4::TransactionInfoMarshaller, | 672 |
| 3054 | decaf::io::ByteArrayInputStream, 834 |
| activemq::wireformat::openwire::marshal::v5::TransactionInfoMarshaller, | decaf::io::ByteArrayOutputStream, 840 |
| 3042 | decaf::io::FilterInputStream, 1534 |
| TransactionState | decaf::io::FilterOutputStream, 1540 |
| activemq::core::ActiveMQConstants, 229 | decaf::net::SocketInputStream, 2801 |
| activemq::state::TransactionState, 3061 | decaf::net::SocketOutputStream, 2806 |
| transfer | decaf::util::AbstractCollection, 129 |
| decaf::internal::util::concurrent::TransferQueue, | decaf::util::concurrent::ConcurrentStlMap, |
| 3063 | 1037 |
| decaf::internal::util::concurrent::TransferStack, | decaf::util::concurrent::locks::Lock, 1901, |
| 3065 | 1902 |
| TransferQueue | decaf::util::concurrent::locks::ReentrantLock, |
| decaf::internal::util::concurrent::TransferQueue, | 2531, 2532 |
| 3063 | decaf::util::concurrent::Mutex, 2241 |
| TransferStack | decaf::util::concurrent::Synchronizable, |
| decaf::internal::util::concurrent::TransferStack, | 2935 |
| 3065 | decaf::util::StlMap, 2855 |
| TransportFilter | decaf::util::StlQueue, 2863 |
| activemq::transport::TransportFilter, 3075 | trylock |
| transportInterrupted | decaf::internal::util::concurrent::MutexImpl, |
| activemq::core::ActiveMQConnection, 211 | 2245 |
| activemq::transport::DefaultTransportListener, | type |
| 1385 | activemq::commands::JournalTransaction, |
| activemq::transport::failover::FailoverTransportListener, | 1793 |
| 1526 | activemq::commands::Message, 2035 |
| activemq::transport::TransportFilter, 3080 | activemq::commands::TransactionInfo, |
| activemq::transport::TransportListener, | 3040 |
| 3082 | |
| transportResumed | UnknownHostException |
| activemq::core::ActiveMQConnection, 211 | decaf::net::UnknownHostException, 3085, |
| activemq::transport::DefaultTransportListener, | 3086 |
| 1385 | UnknownServiceException |
| activemq::transport::failover::FailoverTransportListener, | decaf::net::UnknownServiceException, |
| 1526 | 3088, 3089 |
| activemq::transport::TransportFilter, 3080 | unlock |
| activemq::transport::TransportListener, | activemq::core::MessageDispatchChannel, |
| 3082 | 2093 |
| tryAcquire | decaf::internal::io::StandardErrorOutputStream, |
| | 2815 |

- decaf::internal::io::StandardInputStream, 2823
- decaf::internal::io::StandardOutputStream, 2828
- decaf::internal::util::concurrent::MutexImpl, 2246
- decaf::internal::util::concurrent::SynchronizableImpl, 2944
- decaf::io::BlockingByteArrayInputStream, 672
- decaf::io::ByteArrayInputStream, 834
- decaf::io::ByteArrayOutputStream, 840
- decaf::io::FilterInputStream, 1534
- decaf::io::FilterOutputStream, 1540
- decaf::net::SocketInputStream, 2801
- decaf::net::SocketOutputStream, 2806
- decaf::util::AbstractCollection, 129
- decaf::util::concurrent::ConcurrentStlMap, 1038
- decaf::util::concurrent::Lock, 1904
- decaf::util::concurrent::locks::Lock, 1903
- decaf::util::concurrent::locks::ReentrantLock, 2532
- decaf::util::concurrent::Mutex, 2242
- decaf::util::concurrent::Synchronizable, 2936
- decaf::util::StlMap, 2855
- decaf::util::StlQueue, 2864
- unmarshal
 - activemq::wireformat::openwire::marshal::PrimitiveTypeMarshaller, 2393, 2394
 - activemq::wireformat::openwire::OpenWireFormat, 2308
 - activemq::wireformat::openwire::utils::BooleanStream, 2832
 - activemq::wireformat::stomp::StompWireFormat, 2885
 - activemq::wireformat::WireFormat, 3151
- unmarshalPrimitive
 - activemq::wireformat::openwire::marshal::PrimitiveTypeMarshaller, 2394
- unmarshalPrimitiveList
 - activemq::wireformat::openwire::marshal::PrimitiveTypeMarshaller, 2394
- unmarshalPrimitiveMap
 - activemq::wireformat::openwire::marshal::PrimitiveTypeMarshaller, 2395
- unpark
 - decaf::util::concurrent::locks::LockSupport, 1907
- unregisterFactory
 - activemq::transport::TransportRegistry, 3084
- activemq::wireformat::WireFormatRegistry, 3185
- unregisterSecurityProvider
 - decaf::security_-provider::SecurityProviderMap, 2649
- unsetenv
- decaf::lang::System, 2955
- UNSUBSCRIBE
 - activemq::wireformat::stomp::StompCommandConstants, 2873
- unsubscribe
 - activemq::cmsutil::PooledSession, 2363
 - activemq::core::ActiveMQSession, 417
 - cms::Session, 2675
- UnsupportedEncodingException
 - decaf::io::UnsupportedEncodingException, 3091, 3092
- UnsupportedOperationException
 - cms::UnsupportedOperationException, 3093
 - decaf::lang::exceptions::UnsupportedOperationException, 3094, 3095
- URI
 - decaf::net::URI, 3099, 3100
- URIEncoderDecoder
 - decaf::internal::net::URIEncoderDecoder, 3108
- URIHelper
 - decaf::internal::net::URIHelper, 3112
- URIPool
 - activemq::transport::failover::URIPool, 3119
- URISyntaxException
 - decaf::net::URISyntaxException, 3123,
- URIType
 - decaf::internal::net::URIType, 3128
- URITypesMarshaller
 - decaf::net::URL, 3135
- userID
 - activemq::commands::Message, 2035
- userName
 - activemq::commands::ConnectionInfo, 1134
- UTFDataFormatException

- decaf::io::UTFDataFormatException, 3139, 3140
- UUID
 - decaf::util::UUID, 3143
- validate
 - decaf::internal::net::URLEncoderDecoder, 3109
- validateAuthority
 - decaf::internal::net::URIHelper, 3114
- validateFragment
 - decaf::internal::net::URIHelper, 3115
- validatePath
 - decaf::internal::net::URIHelper, 3115
- validateQuery
 - decaf::internal::net::URIHelper, 3115
- validateScheme
 - decaf::internal::net::URIHelper, 3116
- validateSimple
 - decaf::internal::net::URLEncoderDecoder, 3109
- validateSsp
 - decaf::internal::net::URIHelper, 3116
- validateUserInfo
 - decaf::internal::net::URIHelper, 3116
- value
 - activemq::commands::BrokerId, 694
 - activemq::commands::ConnectionId, 1109
 - activemq::commands::ConsumerId, 1194
 - activemq::commands::LocalTransactionId, 1878
 - activemq::commands::ProducerId, 2450
 - activemq::commands::SessionId, 2681
- valueOf
 - decaf::lang::Boolean, 678
 - decaf::lang::Byte, 783, 784
 - decaf::lang::Character, 921
 - decaf::lang::Double, 1451
 - decaf::lang::Float, 1553, 1554
 - decaf::lang::Integer, 1665, 1666
 - decaf::lang::Long, 1946, 1947
 - decaf::lang::Short, 2737
 - decaf::util::concurrent::TimeUnit, 3013
- values
 - decaf::util::concurrent::ConcurrentStlMap, 1038
 - decaf::util::concurrent::TimeUnit, 3014
 - decaf::util::Map, 1981
 - decaf::util::StlMap, 2856
- variant
 - decaf::util::UUID, 3147
- verify
 - decaf::security::cert::Certificate, 903, 904
 - decaf::security::_-
 - provider::unix::openssl::OpenSSLX509Certificate, 2295
- version
 - decaf::util::UUID, 3147
- visit
 - activemq::commands::BrokerError, 689
 - activemq::commands::BrokerInfo, 720
 - activemq::commands::Command, 995
 - activemq::commands::ConnectionControl, 1058
 - activemq::commands::ConnectionError, 1082
 - activemq::commands::ConnectionInfo, 1133
 - activemq::commands::ConsumerControl, 1169
 - activemq::commands::ConsumerInfo, 1221
 - activemq::commands::ControlCommand, 1246
 - activemq::commands::DestinationInfo, 1394
 - activemq::commands::FlushCommand, 1575
 - activemq::commands::KeepAliveInfo, 1815
 - activemq::commands::Message, 2034
 - activemq::commands::MessageAck, 2059
 - activemq::commands::MessageDispatch, 2088
 - activemq::commands::MessageDispatchNotification, 2119
 - activemq::commands::MessagePull, 2206
 - activemq::commands::ProducerAck, 2423
 - activemq::commands::ProducerInfo, 2473
 - activemq::commands::RemoveInfo, 2539
 - activemq::commands::RemoveSubscriptionInfo, 2563
 - activemq::commands::ReplayCommand, 2586
 - activemq::commands::Response, 2614
 - activemq::commands::SessionInfo, 2704
 - activemq::commands::ShutdownInfo, 2759
 - activemq::commands::TransactionInfo, 3040
 - activemq::commands::WireFormatInfo, 3162
- wait
 - activemq::core::MessageDispatchChannel, 2094, 2095
 - decaf::internal::io::StandardErrorOutputStream, 2815, 2816
 - decaf::internal::io::StandardInputStream, 2823, 2824

- decaf::internal::io::StandardOutputStream, 2828, 2829
- decaf::internal::util::concurrent::ConditionImpl, 1049
- decaf::internal::util::concurrent::SynchronizableImpl, 2944, 2945
- decaf::io::BlockingByteArrayInputStream, 672, 673
- decaf::io::ByteArrayInputStream, 834, 835
- decaf::io::ByteArrayOutputStream, 841
- decaf::io::FilterInputStream, 1535, 1536
- decaf::io::FilterOutputStream, 1541
- decaf::net::SocketInputStream, 2802, 2803
- decaf::net::SocketOutputStream, 2807
- decaf::util::AbstractCollection, 130
- decaf::util::concurrent::ConcurrentStlMap, 1038, 1039
- decaf::util::concurrent::Mutex, 2242, 2243
- decaf::util::concurrent::Synchronizable, 2937, 2938, 2940
- decaf::util::StlMap, 2856, 2857
- decaf::util::StlQueue, 2864, 2865
- WAIT_INFINITE
 - Concurrent.h, 3620
- waitForSpace
 - activemq::util::MemoryUsage, 2017
 - activemq::util::Usage, 3138
- wakeup
 - activemq::core::ActiveMQSession, 418
 - activemq::core::ActiveMQSessionExecutor, 421
 - activemq::threads::CompositeTaskRunner, 1018
 - activemq::threads::DedicatedTaskRunner, 1383
 - activemq::threads::TaskRunner, 2959
- Warn
 - decaf::util::logging, 117
- warn
 - decaf::util::logging::Logger, 1916
 - decaf::util::logging::SimpleLogger, 2785
- wasPrepared
 - activemq::commands::JournalTransaction, 1793
- what
 - cms::CMSException, 963
 - decaf::lang::Exception, 1482
- windowSize
 - activemq::commands::ProducerInfo, 2474
- WireFormatInfo
 - activemq::commands::WireFormatInfo, 3155
- WireFormatInfoMarshaller
 - activemq::wireformat::openwire::marshal::v1::WireFormatInfo, 3172
 - activemq::wireformat::openwire::marshal::v2::WireFormatInfo, 3164
 - activemq::wireformat::openwire::marshal::v3::WireFormatInfo, 3180
 - activemq::wireformat::openwire::marshal::v4::WireFormatInfo, 3176
 - activemq::wireformat::openwire::marshal::v5::WireFormatInfo, 3168
- WireFormatNegotiator
 - activemq::wireformat::WireFormatNegotiator, 3183
- wrap
 - decaf::nio::ByteBuffer, 873
 - decaf::nio::CharBuffer, 945, 946
 - decaf::nio::DoubleBuffer, 1470
 - decaf::nio::FloatBuffer, 1572
 - decaf::nio::IntBuffer, 1651, 1652
 - decaf::nio::LongBuffer, 1966
 - decaf::nio::ShortBuffer, 2756
- write
 - activemq::io::LoggingOutputStream, 1919, 1920
 - decaf::internal::io::StandardErrorOutputStream, 2817
 - decaf::internal::io::StandardOutputStream, 2829, 2830
 - decaf::internal::util::ByteArrayAdapter, 805
 - decaf::io::BufferedOutputStream, 754
 - decaf::io::ByteArrayOutputStream, 842, 843
 - decaf::io::DataOutputStream, 1302, 1303
 - decaf::io::FilterOutputStream, 1542, 1543
 - decaf::io::OutputStream, 2317, 2318
 - decaf::io::Writer, 3187
 - decaf::net::SocketOutputStream, 2808
- writeBoolean
 - activemq::commands::ActiveMQBytesMessage, 178
 - activemq::commands::ActiveMQStreamMessage, 432
 - activemq::wireformat::openwire::utils::BooleanStream, 683
 - cms::BytesMessage, 883
 - cms::StreamMessage, 2900
 - decaf::io::DataOutputStream, 1303
- writeByte
 - activemq::commands::ActiveMQBytesMessage, 178
 - activemq::commands::ActiveMQStreamMessage, 432
 - cms::BytesMessage, 884

- cms::StreamMessage, 2900
- decaf::io::DataOutputStream, 1303
- decaf::io::Writer, 3188
- writeBytes
 - activemq::commands::ActiveMQBytesMessage, 178, 179
 - activemq::commands::ActiveMQStreamMessage, 432, 433
 - cms::BytesMessage, 884
 - cms::StreamMessage, 2901
 - decaf::io::DataOutputStream, 1304
- writeChar
 - activemq::commands::ActiveMQBytesMessage, 179
 - activemq::commands::ActiveMQStreamMessage, 433
 - cms::BytesMessage, 885
 - cms::StreamMessage, 2901
 - decaf::io::DataOutputStream, 1304
- writeChars
 - decaf::io::DataOutputStream, 1304
- WriteChecker
 - activemq::transport::inactivity::InactivityMonitor, 1627
 - activemq::transport::inactivity::WriteChecker, 3186
- writeDouble
 - activemq::commands::ActiveMQBytesMessage, 179
 - activemq::commands::ActiveMQStreamMessage, 433
 - cms::BytesMessage, 885
 - cms::StreamMessage, 2902
 - decaf::io::DataOutputStream, 1305
- writeFloat
 - activemq::commands::ActiveMQBytesMessage, 179
 - activemq::commands::ActiveMQStreamMessage, 434
 - cms::BytesMessage, 885
 - cms::StreamMessage, 2902
 - decaf::io::DataOutputStream, 1305
- writeInt
 - activemq::commands::ActiveMQBytesMessage, 180
 - activemq::commands::ActiveMQStreamMessage, 434
 - cms::BytesMessage, 886
 - cms::StreamMessage, 2902
 - decaf::io::DataOutputStream, 1305
- writeLock
 - decaf::util::concurrent::locks::ReadWriteLock, 2524
- writeLong
 - activemq::commands::ActiveMQBytesMessage, 180
 - activemq::commands::ActiveMQStreamMessage, 434
 - cms::BytesMessage, 886
 - cms::StreamMessage, 2903
 - decaf::io::DataOutputStream, 1305
- writeShort
 - activemq::commands::ActiveMQBytesMessage, 180
 - activemq::commands::ActiveMQStreamMessage, 434
 - cms::BytesMessage, 886
 - cms::StreamMessage, 2903
 - decaf::io::DataOutputStream, 1306
- writeString
 - activemq::commands::ActiveMQBytesMessage, 181
 - activemq::commands::ActiveMQStreamMessage, 435
 - activemq::wireformat::openwire::utils::OpenwireStringSupport, 2316
 - cms::BytesMessage, 887
 - cms::StreamMessage, 2903
- writeTo
 - decaf::io::ByteArrayOutputStream, 843
- writeUnsignedShort
 - activemq::commands::ActiveMQBytesMessage, 181
 - activemq::commands::ActiveMQStreamMessage, 435
 - cms::BytesMessage, 887
 - cms::StreamMessage, 2903
 - decaf::io::DataOutputStream, 1306
- writeUTF
 - activemq::commands::ActiveMQBytesMessage, 181
 - cms::BytesMessage, 887
 - decaf::io::DataOutputStream, 1306
- written
 - decaf::io::DataOutputStream, 1307
- XATransactionId
 - activemq::commands::XATransactionId, 3194
- XATransactionIdMarshaller
 - activemq::wireformat::openwire::marshal::v1::XATransactionId, 3210
 - activemq::wireformat::openwire::marshal::v2::XATransactionId, 3198
 - activemq::wireformat::openwire::marshal::v3::XATransactionId, 3202
 - activemq::wireformat::openwire::marshal::v4::XATransactionId, 3206

activemq::wireformat::openwire::marshal::v5::XATransactionIdMarshaller,
3214